AD-A281 222

The Pennsylvania State University
APPLIED RESEARCH LABORATORY
P.O. Box 30
State College, PA 16804

WEIGHTED PARZEN WINDOWS FOR PATTERN CLASSIFICATION

by

G. A. Babich
L. H. Sibul

DTIC
ELECTE
JUL 07 1994
S B D

Technical Report No. TR 94-10
May 1994

Supported by:
Office of Naval Research

L.R. Hettche, Director
Applied Research Laboratory

Approved for public release; distribution unlimited

94-20511

94 7 6 018

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1994	

4. TITLE AND SUBTITLE

Weighted Parzen Windows for Pattern Classification

5. FUNDING NUMBERS

N00014-90-J-1365

6. AUTHOR(S)

G. A. Babich, L. H. Sibul

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Applied Research Laboratory
The Pennsylvania State University
P.O. Box 30
State College, PA 16804

8. PERFORMING ORGANIZATION REPORT NUMBER

TR#94-10

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
Ballston Tower 1
800 N. Quincy St.
Arlington, Va 22217-5660

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This thesis presents a novel pattern recognition approach, named Weighted Parzen Windows (WPW). This technique uses a nonparametric supervised learning algorithm to estimate the underlying density function for each set of training data. Classification is accomplished by using the estimated density functions in a minimum risk strategy. The proposed approach reduces the effective size of the training data without introducing significant classification error. Furthermore, it is shown that Bayes-Gaussian, minimum Euclidean-distance, Parzen-window, and nearest-neighbor classifiers can be viewed as special cases of the WPW technique. Experimental results are presented to demonstrate the performance of the WPW algorithm as compared to traditional classifiers.

14. SUBJECT TERMS

Parzen Windows, weighted, pattern, recognition, classification, algorithm

15. NUMBER OF PAGES

64

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED

ABSTRACT

This thesis presents a novel pattern recognition approach, named Weighted Parzen Windows (WPW). This technique uses a nonparametric supervised learning algorithm to estimate the underlying density function for each set of training data. Classification is accomplished by using the estimated density functions in a minimum risk strategy. The proposed approach reduces the effective size of the training data without introducing significant classification error. Furthermore, it is shown that Bayes-Gaussian, minimum Euclidean-distance, Parzen-window, and nearest-neighbor classifiers can be viewed as special cases of the WPW technique. Experimental results are presented to demonstrate the performance of the WPW algorithm as compared to traditional classifiers.

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY OF SYMBOLS

c	Number of classes of multidimensional feature data.
ω_i	Label of the ith class where $i = 1, \ldots, c$.
α_i	The decision that an unlabeled sample belongs to the class labeled ω_i.
$g_i(\mathbf{x})$	Discriminant function for the ith class.
$\tilde{g}_i(\mathbf{x})$	Discriminant function that has been changed in some way.
X_i	The set of training samples for the ith class.
n_i	The number of training samples in the ith class.
$\mathbf{x}_{ij}$	The jth training sample of the ith class.
d	Dimensionality of the feature data.
x_k	The kth component of the vector $\mathbf{x}_{ij}$, $k = 1, \ldots, d$.
R_i	The set of reference vectors of the ith class.
$\hat{n}_i$	The number of reference vectors in the ith class.
$\mathbf{r}_{ij}$	The jth reference vector of the ith class.
$\mathbf{w}_i$	Vector containing the window weights of the ith class.
w_{ij}	The jth window weight of the ith class.
e	Average percent error for a single training step.
e_{max}	Maximum desired average percent error for any training step.
$p(\mathbf{x})$	Multivariate continuous probability density function.
$P(\omega_i)$	The a priori probability of the ith class.
$p_n(\mathbf{x})$	Parzen-window multivariate continuous probability density estimate.
$\hat{p}(\mathbf{x})$	Weighted-Parzen-window continuous probability density estimate.
$p(\mathbf{x}\|\omega_i)$	Conditional continuous multivariate probability density function.
$P(\omega_i\|\mathbf{x})$	Discrete a posteriori probability mass function.

$\lambda(\alpha_i\|\omega_j)$	Conditional loss function.
$\rho(\alpha_i\|\mathbf{x})$	Conditional risk function.
$\varphi(\cdot)$	Window function for Parzen-window estimation.
h_i	Smoothing parameter for the ith class.
σ_i	Standard deviation or smoothing parameter for the ith class.
μ_i	Mean vector of the ith class.
Σ_i	Covariance matrix of the ith class.
$\hat{\mu}_i$	Sample mean vector of the ith class.
$\hat{\Sigma}_i$	Sample covariance matrix of the ith class.
$d(\mathbf{x}, \mu_i)$	Euclidean distance from the sample $\mathbf{x}$ to the ith mean.
$d_M(\mathbf{x}, \mu_i)$	Mahalanobis distance from the sample $\mathbf{x}$ to the ith mean.
e_q	Quantization error.
$\mathfrak{R}$	Region of d-dimensional space.
V	Volume enclosed by a surface.
ν	Number of serial computer calculations.
I	The identity matrix.
y	Dummy variable.
$\max_j (x_j)$	Maximum value of j samples.
$\|\cdot\|$	Euclidean distance.
$\|\cdot\|$	Determinant of a matrix or absolute value of a scalar function.
$\equiv$	Defining equation.
iff	If and only if.
Δ	Small incremental value.
$O(\cdot)$	The order of serial calculations necessary.

CHAPTER 1
INTRODUCTION

Pattern recognition techniques can be classified into three groups which are syntactic, neural, and statistical [22:2]. Syntactical pattern recognition techniques analyze the structure of patterns. This approach is used in such areas as model-based vision and artificial intelligence [22:19]. Artificial Neural Networks (ANNs) are mathematical models of biological neurons. ANNs have been employed in pattern recognition tasks such as optical character recognition [11:27]. Statistical pattern recognition covers a wide variety of algorithms whose fundamental building blocks are statistics and probability. This thesis mainly discusses statistical techniques.

Statistical pattern recognition can be subdivided into two groups called *parametric* and *nonparametric*. Parametric techniques assume an underlying probability density function [5:85], [22:66]. Probability models are generally described by *parameters*; therefore, when techniques make use of probability models, they are called parametric. Nonparametric techniques, on the other hand, do not assume a probability model. They, like parametric techniques, however, may also require parameters. Nonparametric techniques can ignore probability models altogether, or in some cases they can be used to estimate a probability model.

The Bayesian parametric technique is the best possible approach since its performance is optimal [5:17], [18:45]. However, it requires a priori knowledge of the probability model which is rarely if ever known in practical applications [5:44]. On the other hand, nonparametric techniques do not require probability models. An advantage of some nonparametric techniques is that they asymptotically approach the true density. Therefore, Bayes optimal performance can be approached. But, nonparametric techniques

often require large sample sets, which, in turn, require significant storage resources [5:122]. Reducing storage requirements without adversely affecting classification performance is a classic problem in statistical pattern recognition. Many approaches have been proposed in statistical pattern recognition literature. P. E. Hart proposed a condensed nearest-neighbor rule in 1968 [12]. In 1972, D. L. Wilson analyzes a nearest-neighbor rule which uses an edited data set [25]. K. Fukunaga et al., addressed the storage problem with a nonparametric data reduction technique in 1984 [7], and they proposed a reduced Parzen classifier in 1989 [8]. More recently, Radial Basis Function (RBF) neural network training paradigms have been proposed to reduce the number of training samples [3], [16], [17]. In this thesis, a novel approach is presented which allows direct control over storage reduction via a single system parameter.

This thesis presents a pattern recognition system called Weighted Parzen Windows (WPW). Statistical pattern recognition techniques and concepts are central to the development and analysis of the WPW approach. Chapter 2 reviews statistical pattern recognition. Chapter 3 presents the WPW algorithm and Chapter 4 presents analytical results of special case training scenarios. Chapter 5 presents experimental results, and Chapter 6 presents a summary and conclusions.

CHAPTER 2
STATISTICAL PATTERN RECOGNITION

In this chapter, several pattern recognition concepts and techniques are reviewed. They are fundamental to the formulation and subsequent analysis of the weighted-Parzen-window technique. Key pattern recognition techniques such as training, classification, and discriminant functions are discussed in the following sections. Also, traditional parametric and nonparametric pattern recognition techniques are presented. They are the Bayes, minimum distance, Parzen window, and k-nearest-neighbor classifiers.

Training

A typical pattern recognition system uses sensors to measure a *state of nature*. Once a measurement is made, a *feature extractor* is used to extract *features* [5:2]. A feature can be any quantity which provides a meaningful description of the state of nature. Throughout this thesis, the data are assumed to be feature data. Feature extraction is discussed in references [5], [9], and [15]. Features are organized into multidimensional *feature vectors* -- $\mathbf{x}$. Feature vectors are also called *sample vectors*. Once the feature data is acquired, *training* can begin. This is a fundamental concept in pattern recognition. Training, often referred to as *learning*, is the process of extracting information from a *training set* of feature data. Techniques can be extremely complicated or simple. For example, some neural paradigms may require billions of computer operations sometimes taking hours or even days to extract information. On the other hand, some statistical techniques may only require learning the mean of the training set. The simplest form of learning is employed by some techniques whose training phase consists of storing a

training set. This can be likened to rote memorization. Although there are many training algorithms, they all perform the same task. They extract information from the training set. This information is used to assign class membership to unlabeled samples, which is called *classification*.

Classification by Discriminant Analysis

Classification is often accomplished by *discriminant analysis*. In a discriminant analysis, discriminant functions are used for each *class* of data which are labeled ω_i where $i = 1, \ldots, c$. Discriminant functions are scalar-valued vector functions denoted as $g_i(\mathbf{x})$ where $i = 1, \ldots, c$. Classification is traditionally accomplished by assigning the class label of largest valued discriminant function to an unlabeled feature vector [5:17], [18:6]. This type of analysis leads to decision boundaries where $g_i(\mathbf{x}) = g_j(\mathbf{x})$ for $i \neq j$. In the case that an unlabeled sample falls on a decision boundary, it is usually assigned class membership by some convenient and arbitrary rule. Discriminant analysis is implemented by a *pattern classifier* or simply a *classifier*.

Properties of Discriminant Functions

Since a pattern classifier compares all discriminant function outputs to find the maximum, only their relative values are important. Therefore, equivalent changes can be made to each discriminant function without affecting the classification results. In other words, the decision boundaries are not changed. Some mathematical operations that do not change classification results are multiplication or division by a positive constant, addition or deletion of a bias, replacement of $g_i(\mathbf{x})$ by $f(g_i(\mathbf{x}))$, where f is a monotonically

increasing function [5:17]. As long as each discriminant function is changed in the same way, the classification results will not be changed.

The Bayes Decision Strategy

The Bayes decision strategy yields a classifier that is optimal, i.e., the classification error rate is minimal. This concept is paramount to pattern recognition regardless of the technique used. Therefore, a detailed discussion will follow. First, Bayes rule will be presented. Then, the concept of conditional risk will be used to derive the optimal classifier. Finally, the Gaussian multivariate discriminant function will be presented. The formulation of this section follows that of Duda and Hart [5].

Bayes Rule and Conditional Risk. Let $\mathbf{x}$ be a d component feature vector which obeys the class conditional probability density function $p(\mathbf{x}|\omega_i)$, where ω_i represents one of c possible states of nature that are of interest, and $P(\omega_i)$ represents the a priori probability that ω_i occurs. The conditional a posteriori probability of a state of nature can be expressed by Bayes rule:

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})}, \tag{2.1}$$

where

$$p(\mathbf{x}) = \sum_{i=1}^{c} p(\mathbf{x}|\omega_i)P(\omega_i). \tag{2.2}$$

Now that the a posteriori probability function has been found by Bayes rule, the next step is to define a suitable loss function. Given the action α_i and the true state of

nature is ω_j, the conditional loss is given by $\lambda(\alpha_i|\,\omega_j)$. The conditional risk or expected loss of taking the action α_i is given by

$$\rho(\alpha_i|\mathbf{x}) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j) P(\omega_j|\mathbf{x}). \tag{2.3}$$

The optimal decision rule is one that achieves the minimal overall conditional risk. Such a decision rule can be achieved by taking the action that minimizes the overall risk [5:17], [18:45], [15:188-190] as given by Equation (2.3). A commonly used loss function that minimizes the overall risk is the symmetrical or zero-one loss function [5:16]. This function assigns a unit loss when the action α_i is taken and the actual state of nature is ω_j, if and only if $i \neq j$. Stated mathematically

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases} \qquad i,j = 1,\ldots,c \tag{2.4}$$

Using Equation (2.4), the optimal decision rule can be derived. The conditional risk, given by Equation (2.3), can be simplified by substituting Equation (2.4) for the loss function. Since the sum of the conditional probability mass functions $P(\omega_j \mid \mathbf{x})$ taken over all j must equal 1, the conditional risk can be simplified as shown:

$$\rho(\alpha_i|\mathbf{x}) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j) P(\omega_j|\mathbf{x})$$

$$= \sum_{\substack{j=1 \\ j\neq i}}^{c} P(\omega_j|\mathbf{x})$$

$$= 1 - P(\omega_i|\mathbf{x}). \tag{2.5}$$

With this result it is clear that the minimal error rate can be achieved by selecting the class i that minimizes Equation (2.5), i.e., the class with the largest a posteriori conditional probability.

Discriminant Analysis with Bayes Rule. Equation (2.5) is especially suited for discriminant analysis. Since the error rate is minimal when choosing the class with the largest conditional probability, one needs only to calculate $P(\omega_i \mid \mathbf{x})$ for each class i and label the test sample according to the largest value. In other words, the optimal decision rule with the smallest possible error [5:17] is given as:

Decide that $\mathbf{x}$ belongs to class ω_i iff $\quad P(\omega_i \mid \mathbf{x}) > P(\omega_j \mid \mathbf{x})$ for all $j \neq i$. (2.6)

The decision rule is also given by:

Decide that $\mathbf{x}$ belongs to class ω_i iff $\quad g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$ (2.7)

where, by Bayes rule

$$g_i(\mathbf{x}) = P(\omega_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_i) P(\omega_i)}{p(\mathbf{x})},$$

which can be simplified by removing the scaling constant $p(\mathbf{x})$:

$$\tilde{g}_i(\mathbf{x}) = p(\mathbf{x} \mid \omega_i) P(\omega_i). \tag{2.8}$$

The tilde notation of Equation (2.8) indicates that the discriminant function has been changed, but the classification results remain the same.

The Bayes-Gaussian Discriminant Function. In what follows, the Bayes-Gaussian decision rule is derived. It will be shown that the resulting discriminant function is quadratic in the most general case and linear when certain assumptions are made.

The discriminant function is derived by first eliminating the scaling factor $p(\mathbf{x})$ from Equation (2.8) since it is common to all classes [5:17-18]. The discriminant function is given as

$$g_i(\mathbf{x}) = p(\mathbf{x} \mid \omega_i) P(\omega_i). \tag{2.9}$$

The Gaussian multivariate conditional density for the ith class in d-dimensional space is given by

$$p(\mathbf{x}|\omega_i) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_i|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_i)^t \Sigma_i^{-1}(\mathbf{x}-\mu_i)\right] \tag{2.10}$$

where μ_i and Σ_i are the mean vector and covariance matrix respectively, and $|\cdot|$ is the determinant. Substituting Equation (2.10) into Equation (2.9), the discriminant function becomes

$$g_i(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_i|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_i)^t \Sigma_i^{-1}(\mathbf{x}-\mu_i)\right] P(\omega_i). \tag{2.11}$$

By taking the natural log of Equation (2.11), the quadratic discriminant function is obtained. Equation (2.12) shows the general case quadratic discriminant function, which gives the same classification results as the exponential discriminant function.

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\mu_i)^t \Sigma_i^{-1}(\mathbf{x}-\mu_i) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i) \tag{2.12}$$

The quadratic discriminant function can be further reduced to a linear discriminant function when the covariance matrices are equivalent for each class ($\Sigma_i = \Sigma$). In what follows, the linear discriminant function is derived. First, Equation (2.12) is simplified by removing bias terms that are present in each discriminant function. The simplified version is given by Equation (2.13):

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\mu_i)^t \Sigma^{-1}(\mathbf{x}-\mu_i) + \ln P(\omega_i) \tag{2.13}$$

Then, the quadratic term is expanded:

$$g_i(\mathbf{x}) = -\frac{1}{2}\left(\mathbf{x}^t\Sigma^{-1}\mathbf{x} - 2\mu_i{}^t\Sigma^{-1}\mathbf{x} + \mu_i{}^t\Sigma^{-1}\mu_i\right) + \ln P(\omega_i). \tag{2.14}$$

Inspection of Equation (2.14) reveals that the quadratic term, in $\mathbf{x}$, can be removed since it is common to each discriminant function. The linear discriminant function is given by

$$g_i(\mathbf{x}) = \left(\Sigma^{-1}\mu_i\right)^t \mathbf{x} - \frac{1}{2}\mu_i{}^t\Sigma^{-1}\mu_i + \ln P(\omega_i). \tag{2.15}$$

Another linear discriminant function can be derived for the case when $\Sigma_i = \sigma^2 I$, where I is a dxd identity matrix. In this case, the discriminant function is given by

$$g_i(\mathbf{x}) = \left(\frac{1}{\sigma^2}\mu_i\right)^t \mathbf{x} - \frac{1}{2\sigma^2}\mu_i{}^t\mu_i + \ln P(\omega_i). \tag{2.16}$$

This section has shown that the Bayes decision strategy is used to find an optimal classifier. Furthermore, three Bayesian discriminant functions were derived for a Gaussian distribution. The first of these, the arbitrary case, was shown to be quadratic (Equation (2.12)). The decision boundaries for the quadratic case are *hyperquadrics* [5:30]. The second two discriminant functions were derived for the case when covariance matrices are identical for each class. The decision boundaries for each of these cases are *hyperplanes* [5:26-30].

Minimum Distance Classification

Minimum distance classifiers are widely referenced throughout the literature [5], [18], [20], [22]. Quite often the mean or sample mean of a class is used as a prototype. With this type of classifier, unknown feature vectors are assigned the class membership of the nearest mean. Two metrics are commonly used -- Mahalanobis and Euclidean. The following reviews both of these classifiers.

Minimum Mahalanobis-Distance. The squared Mahalanobis distance of a feature vector $\mathbf{x}$ from the ith class mean is given by

$$d_M^2(\mathbf{x},\boldsymbol{\mu}_i) = (\mathbf{x}-\boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) \tag{2.17}$$

where $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the mean vector and covariance matrix respectively. Since class membership is assigned based on the smallest distance given by Equation (2.17), a discriminant function can be written as

$$g_i(\mathbf{x}) = -(\mathbf{x}-\boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i). \tag{2.18}$$

Equation (2.18) bears resemblance to the Bayes-Gaussian discriminant function of Equation (2.12). In fact, the minimum Mahalanobis-distance classifier is optimal in the case of Gaussian distributions with equal covariance matrices and equal a priori probabilities. However, the discriminant function of (2.18) is strictly nonparameteric. That is to say that no underlying distribution is assumed. Therefore, the mean vector and covariance matrices are generally found by samples. Given n_i training samples from the ith class, the sample mean [5:48] is given by

$$\hat{\boldsymbol{\mu}}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} \mathbf{x}_{ij}. \tag{2.19}$$

The sample covariance matrix [5:49] is given by

$$\hat{\boldsymbol{\Sigma}}_i = \frac{1}{n_i - 1}\sum_{j=1}^{n_i}(\mathbf{x}_{ij}-\hat{\boldsymbol{\mu}}_i)(\mathbf{x}_{ij}-\hat{\boldsymbol{\mu}}_i)^t. \tag{2.20}$$

Minimum Euclidean-Distance. The squared Euclidean distance of a feature vector $\mathbf{x}$ from the ith class mean is given by

$$d^2(\mathbf{x},\boldsymbol{\mu}_i) = (\mathbf{x}-\boldsymbol{\mu}_i)^t(\mathbf{x}-\boldsymbol{\mu}_i) \tag{2.21}$$

where $\boldsymbol{\mu}_i$ is the mean vector. Since class membership is assigned based on the smallest

distance given by Equation (2.21), a discriminant function can be written as

$$g_i(\mathbf{x}) = -(\mathbf{x} - \boldsymbol{\mu}_i)^t(\mathbf{x} - \boldsymbol{\mu}_i). \tag{2.22}$$

As with the Mahalanobis classifier, it can be shown that Equation (2.22) is optimal in certain cases, i.e., when $\Sigma_i = \sigma^2 I$. However, the Euclidean classifier is generally nonparametric since no density function is assumed. The mean vector of Equation (2.22) is usually found by Equation (2.19). Below, it is shown that the minimum Euclidean-distance classifier can be implemented by a linear discriminant function. First, Equation (2.22) is expanded. This reveals that $\mathbf{x}^t\mathbf{x}$ is a bias term present in each discriminant function. It is removed and the new discriminant function is denoted by the tilde notation which indicates that classification results are not changed. Equation (2.23) shows the final result which gives the same classification results as Equation (2.22).

$$\begin{aligned} g_i(\mathbf{x}) &= -\mathbf{x}^t\mathbf{x} + 2\boldsymbol{\mu}_i{}^t\mathbf{x} - \boldsymbol{\mu}_i{}^t\boldsymbol{\mu}_i \\ \tilde{g}_i(\mathbf{x}) &= 2\boldsymbol{\mu}_i{}^t\mathbf{x} - \boldsymbol{\mu}_i{}^t\boldsymbol{\mu}_i \\ \tilde{\tilde{g}}_i(\mathbf{x}) &= \boldsymbol{\mu}_i{}^t\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_i{}^t\boldsymbol{\mu}_i \end{aligned} \tag{2.23}$$

As can be seen by Equation (2.23), the minimum Euclidean-distance classifier is linear in $\mathbf{x}$.

The minimum Mahalanobis-distance classifier is quadratic. Therefore, its decision boundaries are hyperquadric surfaces. It has been shown that the Euclidean distance classifier can be implemented as a linear discriminant function; therefore, its decision boundaries are hyperplanes. The performance of the quadratic classifier often suffers due to non-normality of the data; however, the linear classifier is robust to non-normality [20:253].

Parzen-Window Density Estimation and Classification

This is a nonparametric technique that assumes no underlying distribution but estimates a probability density function. In his classic paper "On Estimation of Probability Density Function and Mode," Parzen showed that the density estimate will approach the actual density as the number of training samples approaches infinity [19]. This is true for certain easily met conditions. The density model and conditions necessary for convergence are discussed below.

Let n be the number of samples drawn form a particular distribution $p(x)$. The general form of the probability density estimate $p_n(x)$ in the Parzen-window technique is

$$p_n(x) = \frac{1}{nh}\sum_{j=1}^{n}\varphi\left(\frac{x - x_j}{h}\right), \tag{2.24}$$

where h is a parameter "suitably chosen" [19:1066], x_j is the jth training sample, and $\varphi(y)$ is the window function. Note that the notation above is for a univariate training set. Parzen states that if h is chosen to satisfy a mild restriction as a function of n, the estimate $p_n(x)$ is asymptotically unbiased, or in mathematical terms if

$$\lim_{n\to\infty} h(n) = 0, \tag{2.25}$$

and,

$$\lim_{n\to\infty} nh(n) = \infty, \tag{2.26}$$

then

$$\lim_{n\to\infty} E[p_n(x)] = p(x). \tag{2.27}$$

Parzen also shows that the window function $\varphi(y)$ must satisfy the following requirements:

$$\sup_{-\infty<y<\infty} |\varphi(y)| < \infty \tag{2.28}$$

$$\int_{-\infty}^{\infty}|\varphi(y)|dy < \infty \tag{2.29}$$

$$\lim_{y\to\infty}|y\varphi(y)| = 0 \tag{2.30}$$

$$\int_{-\infty}^{\infty}\varphi(y)dy = 1 \tag{2.31}$$

where $|\cdot|$ is the absolute value.

A popular window shape is Gaussian. The density estimate for a multivariate Gaussian window is given by

$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\left(2\pi\sigma^2\right)^{\frac{d}{2}}}\exp\left[-\frac{1}{2\sigma^2}(\mathbf{x}-\mathbf{x}_i)'(\mathbf{x}-\mathbf{x}_i)\right] \tag{2.32}$$

where $\mathbf{x}_i$ is the ith training sample, d is the number of dimensions, and n is the number of training samples in the ith class. Note that σ replaces h to emphasize the relation to the Gaussian density function. Equation (2.33) shows the discriminant function form of the Parzen estimate in a Bayes strategy.

$$g_i(\mathbf{x}) = P(\omega_i)\frac{1}{n_i}\sum_{j=1}^{n_i}\frac{1}{\left(2\pi\sigma_i^2\right)^{\frac{d}{2}}}\exp\left[-\frac{1}{2\sigma_i^2}(\mathbf{x}-\mathbf{x}_{ij})'(\mathbf{x}-\mathbf{x}_{ij})\right] \tag{2.33}$$

The finite sample case of the Parzen-window classifier is not generally optimal. In these cases, selection of the h parameter, often called the smoothing parameter, greatly affects the classifier's performance. Selection of the smoothing parameter is discussed throughout the literature [5], [9], [19], [10], [20]. Due to the intractable nature of an analytical solution, experimental approaches are generally used to find the appropriate

smoothing parameter. Reference [20:254] suggests a technique in which several smoothing parameters are tested simultaneously to find the best choice.

The *k*-Nearest-Neighbor Rule

The *k*-Nearest-Neighbor (*k*NN) technique is nonparametric, assuming nothing about the distribution of the data. Stated succinctly, this rule assigns the class membership of an unlabeled pattern to the same class as its *k*-nearest training patterns. In the case that not all the neighbors are from the same class, a voting scheme is used. Duda and Hart [5:104] state that this rule can be viewed as an estimate of "the a posteriori probabilities $P(\omega_i \mid \mathbf{x})$ from samples." Raudys and Jain [20:255] advance this interpretation by pointing out that the *k*NN technique can be viewed as the "Parzen window classifier with a hyper-rectangular window function." As with the Parzen-window technique, the *k*NN classifier is more accurate as the number of training samples increases [5:105].

A special case of the *k*NN technique is when $k = 1$. This case, known as the NN classifier, was studied in detail by Cover and Hart [4], who showed that its performance was bounded by twice the Bayes error rate in the "large sample case." The NN rule can be stated in discriminant function form as

$$g_i(\mathbf{x}) = \max_j \left[-(\mathbf{x} - \mathbf{x}_{ij})^t (\mathbf{x} - \mathbf{x}_{ij}) \right] \tag{2.34}$$

where $\mathbf{x}_{ij}$ is the jth training sample of the class labeled ω_i, and $\mathbf{x}$ denotes the unknown test pattern. A simplified version of this discriminant function is given by

$$g_i(\mathbf{x}) = \max_j \left(\mathbf{x}_{ij}{}^t \mathbf{x} - \frac{1}{2} \mathbf{x}_{ij}{}^t \mathbf{x}_{ij} \right). \tag{2.35}$$

CHAPTER 3
WEIGHTED PARZEN WINDOWS

This chapter introduces a novel nonparametric pattern recognition approach, named Weighted Parzen Windows (WPW). First, the training and classification algorithms are presented. Then, it is shown that the training algorithm is stepwise optimal. Also, the computational complexity of the training and classification algorithms is discussed. Finally, design considerations are discussed.

Weighted-Parzen-Window Training

Training is a parallel operation in that training for the class labeled ω_i is independent of the training for the class labeled ω_j, for $i \neq j$. Therefore, training can be conducted in parallel. Thus, the following discussion will focus on a single class to simplify notation. The training phase will be presented in algorithmic form followed by a discussion of the major concepts.

Training Algorithm. Given a training set of n d-dimensional feature samples, $X = \{ \mathbf{x}_1, \ldots, \mathbf{x}_n \}$ where $\mathbf{x} = [x_1, \ldots, x_d]^t$ the basic approach of the WPW training algorithm is to find a set of $\hat{n}$ *reference vectors*[1], $R = \{ \mathbf{r}_1, \ldots, \mathbf{r}_{\hat{n}} \}$, where $1 \leq \hat{n} \leq n$. Since the number of samples in R can be less than the number of samples in the original training set, some information may be lost. To compensate for lost information, a set of $\hat{n}$ weights, $\mathbf{w} = \{ w_1, \ldots, w_{\hat{n}} \}$, are found. Each scalar weight w_j corresponds to the

[1] If readers are familiar with classical Vector Quantization (VQ), they will recognize that a collection of reference vectors is the same as a *codebook*. Current research has focused on pattern recognition, although the training algorithm is directly applicable to VQ applications. Excellent treatment of classical VQ can be found in reference [1].

reference vector $r_j, j = 1, \ldots, \hat{n}$. The role of the weights is discussed in the sequel. The training algorithm for a single class is presented in Table 3.1. In Table 3.1, the estimate, $\hat{p}(\mathbf{x})$, is given by Equation (3.1)[2].

$$\hat{p}(\mathbf{x}) = \frac{1}{nh} \sum_{i=1}^{\hat{n}} \varphi\left(\frac{\mathbf{x} - r_i}{h}\right) \cdot w_i \qquad (3.1)$$

Training Concepts. The WPW algorithm can be considered a *second order approximation* since it is relying on the Parzen estimate and, therefore, can only be as accurate as the Parzen estimate. As can be seen by Equation (3.1), the WPW estimate is a superposition of *weighted Parzen windows*. This estimate is a *quantized* version of the Parzen estimate. Quantization occurs when two window functions that are *close* in vector space are combined to create a new single weighted-window function. The new window function is weighted by the total number of combinations that its center has undergone. In other words, w_i window functions are centered at $\mathbf{r}_i$ which is the average vector of w_i similar training samples. This procedure allows the training algorithm to learn the densest regions of the training set, which, in turn, allows reference vector reduction. This reduction is offset by weight adjustments, which allow the algorithm to remember where the densest regions occur. The training algorithm allows for quantization of the vector space with respect to the probability space. In this technique, storage requirements are traded-off for probability space error.

---

[2] Neural network literature often refers to this type of equation as a radial basis function (RBF) [3], [16], [17]. Current research has focused on statistical pattern recognition, although WPW training is directly applicable to RBF neural network design.

Table 3.1: WPW training algorithm for a single class of feature data.

Step 1. Calculate and store $p_n(\mathbf{x}_k)$ where $k = 1, \ldots, n$.

Step 2. Choose: $h > 0$, $e_{max} \geq 0$.

Step 3. Initialize: $\hat{n} \leftarrow n$, $R \leftarrow X$, $w_i = 1$ where $i = 1, \ldots, \hat{n}$.

Step 4. Choose two closest[3] reference vectors $\mathbf{r}_i$ and $\mathbf{r}_j$, $i \neq j$.

Step 5. Calculate the vector $r_o = \dfrac{r_i w_i + r_j w_j}{w_i + w_j}$.

Step 6. (a) update R such that $\{ \mathbf{r}_j, \mathbf{r}_i \} \notin R$ and $\mathbf{r}_o \in R$,

(b) update the coefficient for $\mathbf{r}_o$, $w_o \leftarrow w_i + w_j$,

(c) update $\hat{n} \leftarrow \hat{n} - 1$.

Step 7. Calculate $e = \left[\dfrac{1}{n} \sum_{k=1}^{n} \dfrac{\left| p_n(\mathbf{x}_k) - \hat{p}(\mathbf{x}_k) \right|}{p_n(\mathbf{x}_k)} \right] 100\%$.

Step 8. IF ($e < e_{max}$) THEN,

if $\hat{n} = 1$, then stop training, output R and $\mathbf{w}$;

otherwise, go to Step 4;

ELSEIF ($e > e_{max}$) THEN

reconstruct R such that $\{ \mathbf{r}_i, \mathbf{r}_j \} \in R$ and $\mathbf{r}_O \notin R$,

replace coefficients for $\mathbf{r}_i$, $\mathbf{r}_j$ *as* w_i, w_j,

adjust $\hat{n} \leftarrow \hat{n} + 1$,

stop training, output R and $\mathbf{w}$;

ENDIF

---

[3] The meaning of closest is not discussed in detail in this section. In general, however, the closeness of two reference vectors should be measured with the metric that the window function uses. A detailed discussion can be found in the section which addresses stepwise optimization.

Combination of reference vectors is controlled by the training algorithm. An error function is used to measure the deviation of the new estimate $\hat{p}(\mathbf{x})$ from the Parzen estimate $p_n(\mathbf{x})$ at each of the training samples. The error function in step 7 of the training algorithm (Equation (3.2)) is the average percent error between the two estimates at each training sample:

$$e = \left[\frac{1}{n}\sum_{k=1}^{n}\frac{|p_n(\mathbf{x}_k) - \hat{p}(\mathbf{x}_k)|}{p_n(\mathbf{x}_k)}\right]100\%. \tag{3.2}$$

As long as e is below e_{max}, reference vectors will be combined according to step 5 (Equation (3.3)) of the training algorithm, and $\hat{n}$ will continue to decrease:

$$r_o = \frac{r_i w_i + r_j w_j}{w_i + w_j}.$$

$$= \frac{w_i}{w_i + w_j} r_i + \frac{w_j}{w_i + w_j} r_j. \tag{3.3}$$

When reference vectors are combined, weights are combined according to

$$w_o = w_i + w_j \qquad i \neq j. \tag{3.4}$$

(Note: the training algorithm requires that $\sum_{j=1}^{\hat{n}} w_j = n$.) Clearly, the weights are a method of counting the number of reference vectors combined into a single reference vector, and they are integer values.

Classification Algorithm

Once the reference vectors and weights are found for each category, a discriminant analysis approach can be used. For the reference vectors R_i and the weights $\mathbf{w}_i$, the discriminant function for the ith category is given by

$$g_i(\mathbf{x}) = P(\omega_i)\frac{1}{n_i h_i}\sum_{j=1}^{\hat{n}_i}\varphi\left(\frac{\mathbf{x}-\mathbf{r}_{ij}}{h_i}\right)\cdot w_{ij} \qquad (3.5)$$

where $P(\omega_i)$ is the a priori probability of the class labeled ω_i, n_i is the number of training samples in the ith class, $\hat{n}_i$ is the number of reference vectors in the ith class, $\mathbf{r}_{ij}$ is the jth reference vector of the ith class, w_{ij} is the jth coefficient corresponding to $\mathbf{r}_{ij}$, $\varphi(\cdot)$ is the window function, and h_i is the parameter that controls the window width for the ith category. Equation (3.5) is simply Equation (3.1) weighted by the a priori probability of a given class. This ensures that a Bayes-optimal solution can be approached (see Equation (2.7)). Equation (3.5) is written in a compact form below to show its similarity to the optimal Bayes strategy of Equation (2.8).

$$g_i(\mathbf{x}) \approx \hat{p}_i(\mathbf{x}|\omega_i)P(\omega_i). \qquad (3.6)$$

Pattern classification of multidimensional feature data can be achieved by first training with the algorithm in Table 3.1, then using Equation (3.6) for testing by discriminant function analysis.

Stepwise Optimization

The WPW training algorithm quantizes vector space is based on a probability space error criterion. Quantization causes the WPW estimate to deviate from the Parzen estimate, thereby introducing error between the two. Once this error, as measured by Equation (3.2), exceeds a predetermined value, e_{max}, training is halted. Since the objective of the training algorithm is to reduce the number of reference vectors without introducing error into the density estimate, it makes sense to minimize error for each training step. One way to minimize stepwise error is to minimize *quantization error*. Quantization error is introduced in each step of the training algorithm as a result of

combining two reference vectors. In what follows, it will be proven that the WPW training algorithm minimizes quantization error for each step.

Consider combining two reference vectors $\mathbf{r}_i$ and $\mathbf{r}_j$ whose weights are w_i and w_j respectively. The resulting reference vector $\mathbf{r}_o$ is given by Equation (3.3), and its weight is given by $w_o = w_i + w_j$ (see Equation (3.4)). On the kth training step, $\mathbf{r}_i$ and $\mathbf{r}_j$ are the center of two weighted window functions $w_i\varphi(\cdot)$ and $w_j\varphi(\cdot)$, which contribute to the density given by Equation (3.1). On the $k+1st$ step, after combination, their contribution is a single weighted window function, $w_o\varphi(\cdot)$, centered at $\mathbf{r}_o$. The quantization error introduced by this combination is defined in terms of the three above mentioned weighted window functions. The volume enclosed by each of these weighted windows is denoted by V_i,V_j, and V_o. Given a vector space $\Re$, the region within V_o is denoted as $\Re_o$. The regions of intersection $(V_i \cap V_o)$ and $(V_j \cap V_o)$ are denoted as $\Re_i$ and $\Re_j$ respectively. The quantization error integral is defined as

$$e_q \equiv \frac{1}{w_i + w_j}\left\{(w_i + w_j)\int_{\Re_o}\varphi\left(\frac{\mathbf{x}-\mathbf{r}_o}{h}\right)d\mathbf{x} - \left[w_i\int_{\Re_i}\varphi\left(\frac{\mathbf{x}-\mathbf{r}_i}{h}\right)d\mathbf{x} + w_j\int_{\Re_j}\varphi\left(\frac{\mathbf{x}-\mathbf{r}_j}{h}\right)d\mathbf{x}\right]\right\},$$

but proper selection of the Parzen-window function, $\varphi(\cdot)$, requires that its volume equal 1, so the quantization error is

$$= \frac{1}{w_i + w_j}\left[w_i + w_j - w_i\int_{\Re_i}\varphi\left(\frac{\mathbf{x}-\mathbf{r}_i}{h}\right)d\mathbf{x} - w_j\int_{\Re_j}\varphi\left(\frac{\mathbf{x}-\mathbf{r}_j}{h}\right)d\mathbf{x}\right]$$

$$= \frac{1}{w_i + w_j}\left\{w_i\left[1 - \int_{\Re_i}\varphi\left(\frac{\mathbf{x}-\mathbf{r}_i}{h}\right)d\mathbf{x}\right] + w_j\left[1 - \int_{\Re_j}\varphi\left(\frac{\mathbf{x}-\mathbf{r}_j}{h}\right)d\mathbf{x}\right]\right\}. \qquad (3.7)$$

Maximum quantization error occurs when the integrals of Equation (3.7) are both equal to 0 and results in a quantization error of 1. This is the worse case scenario when $(V_i \cap V_o)$ and $(V_j \cap V_o)$ are equal to zero. Clearly, the minimum quantization error occurs when the integral terms are equal to 1, which results in a quantization error of 0. Equation (3.7) is exactly 0 if and only if $\Re_i$ and $\Re_j$ are completely enclosed within $\Re_o$. But, $\Re_i$ and $\Re_j$ can only be completely enclosed within $\Re_o$ if and only if $w_i\varphi(\cdot)$ and $w_j\varphi(\cdot)$ are completely enclosed within $w_o\varphi(\cdot)$. Geometrically, this is only true if each window function shares the same center, i.e., when

$$\mathbf{r}_i = \mathbf{r}_j = \mathbf{r}_o. \tag{3.8}$$

The best procedure to follow when deciding which two vectors to combine is to select the vectors whose associated regions $\Re_i$ and $\Re_j$ are the largest. To maximize $\Re_i$ and $\Re_j$, Step 4 of the training algorithm selects the two *closest* reference vectors as the two to be combined. The distance measure used should be the same as the measure used by the Parzen-window function. The closest two reference vectors are defined as the two vectors $\mathbf{r}_i$ and $\mathbf{r}_j$ that are closest to their corresponding $\mathbf{r}_o$. Since Equation (3.3) is a convex combination of two vectors, the two reference vectors which are closest to their corresponding $\mathbf{r}_o$ are simply the two closest vectors $\mathbf{r}_i$ and $\mathbf{r}_j$, where $i \neq j$.

Returning to the proof of stepwise optimality, it can be stated that when deciding which two vectors to combine, one should select the two closest as measured by Equation (3.9). By selecting the two closest vectors, the quantization error for a given step is minimized. Or, mathematically

$$\lim_{d_W(\mathbf{r}_i,\mathbf{r}_j)\to 0} e_q = 0 \tag{3.9}$$

where d_W is the distance measure used by the window function. Since the WPW training algorithm uses this procedure, it is stepwise optimal.

Computational Complexity

In this section, the computational complexity of the WPW technique is discussed. As can be seen by the training and classification algorithms, distance calculations require the bulk of processing resources. Therefore, the following analysis will determine the order of magnitude of the distance calculations for a single class of data. The following analysis is based on serial computation.

Training Complexity. Distance calculations are required in three of the WPW training steps. As shown in Table 3.1, distance calculations are required in Steps 1, 3, and 7. In Step 1, n probability calculations are required, each with n distance calculations. Therefore, the number of distance calculations, ν_1, in Step 1 is given by

$$\nu_1 = n^2. \tag{3.10}$$

The number of distance calculations necessary in Step 3 of the training phase is related to the number of reference vectors. On the jth training step, there are k reference vectors bounded by $\hat{n}\text{-}1 \le k \le n$. The lower limit of k is given by $\hat{n}\text{-}1$ because the training algorithm always combines two reference vectors before the error function is calculated. In the case of $\hat{n} = 1$, the training algorithm terminates, so no distance calculations are performed (refer to Step 8 of the training algorithm). Step 3 of the training algorithm requires the calculation of $k(k\text{-}1)$ distance measures for the jth step. The total number of calculations for this step over the entire training phase is given by

$$\nu_2 = \sum_{k=\hat{n}-1}^{n} k(k-1). \tag{3.11}$$

Equation (3.11) can be used to calculate the worse case serial requirements. In the worse case, when $\hat{n} = 1$, ν_{2max} is given by

$$v_{2\max} = \sum_{k=0}^{n} k(k-1)$$

$$= \sum_{k=1}^{n} k^2 - \sum_{k=1}^{n} k$$

$$= \frac{1}{6}[n(n+1)(2n+1)] - \frac{1}{2}n(n+1)$$

$$= \frac{1}{3}(n^3 - n). \tag{3.12}$$

The distance calculations required in Step 7 of the training algorithm are dependent on the value of $\hat{n}$. During this step, there are n probability calculations. Each probability calculation requires k distance calculations where $\hat{n}$-1 ≤ k ≤ n. As explained for Equation (3.11), the lower limit of k is given by $\hat{n}$-1. The total distance computations necessary for Step 7 are given by

$$v_3 = n \sum_{k=\hat{n}-1}^{n} k. \tag{3.13}$$

In the worse case, $v_{3\max}$ is given by

$$v_{3\max} = n \sum_{k=0}^{n} k$$

$$= n \sum_{k=1}^{n} k$$

$$= \frac{1}{2} n^2(n+1)$$

$$= \frac{1}{2}(n^3 + n^2). \tag{3.14}$$

Combining Equations (3.10), (3.12) and (3.14), the total number of distance calculations during training, for the worse case, is given by

$$v_{\text{train}} = O(n^3). \tag{3.15}$$

Classification Complexity. The number of distance calculations required by a single discriminant function is $\hat{n}$ for each test made. The largest $\hat{n}$ can be is n. The upper bound on the number of distance calculations necessary for a single discriminant function is given by

$$\nu_{test} = O(n). \tag{3.16}$$

The computational complexity of the WPW training algorithm requires significantly more distance calculations than the classification algorithm. The number of distance calculations for a single class of data is $O(n^3)$. The number of calculations for all training classes is still $O(n^3)$ because the number of classes is generally much smaller than n. Although calculations of this order can be severe, it must be noted that the above analysis is for worse case serial computation. Possible refinements can be made when finding the two closest vectors in Step 4 of the training algorithm, e.g., preprocessing the reference vectors and recursively updating them on every step. Several quick search routines are available to programmers [6], [9], [23], [26]. It may be possible to modify such a routine for the WPW training algorithm. Also, it may be possible to store the WPW density estimate in a table updating it recursively on each training step to reduce the calculations necessary on Step 7 of training algorithm. Finally, the power of parallel computation can be invoked making WPW training nearly trivial since all distance calculations can be calculated simultaneously. Regardless of the possible shortcuts, the analysis of the training algorithm shows that it will terminate after a finite number of training steps and is at worst $O(n^3)$. The classification algorithm on the other hand is at worst $O(n)$.

Designing the Weighted-Parzen-Window Classifier

Selecting the Window Shape. When selecting a window shape, those suitable for the Parzen density estimate should be chosen. Any window shape satisfying the conditions as established by Parzen [19] are sufficient, and they should be used for both the Parzen, $p_n(\mathbf{x})$ and WPW estimates, $\hat{p}(\mathbf{x})$. Several window shapes can be found in references [5] and [19]. The mostly widely referenced window function is Gaussian.

Selecting the Smoothing Parameter. The smoothing parameter should be the same as that used for Parzen estimate, $p_n(\mathbf{x})$. Choosing the smoothing parameter is a critical step in the classifier design. Selection of the smoothing parameter is discussed throughout the literature [5], [9], [10], [19], [20]. Due to the intractable nature of an analytical solution, experimental approaches are generally used to find the appropriate smoothing parameter. Reference [20:254] suggests a technique in which several smoothing parameters are tested simultaneously to find the best choice. Although the training and classification algorithms allow for selection of different smoothing parameters for each class of data, it is recommended to use a single value for all classes [20:254].

Selecting the Maximum Allowable Error. The value e_{max} is used to control the training algorithm's aggressiveness. That is to say, if e_{max} is small then the number of vectors in R will be nearly n; otherwise, if e_{max} is large, then $\hat{n} << n$. Equation (3.20) shows how $\hat{n}$ is affected by e_{max} in the limit.

$$\lim_{e_{max} \to 0} \hat{n} = n \qquad (3.20.a)$$

$$\lim_{e_{max} \to \infty} \hat{n} = 1 \qquad (3.20.b)$$

The relationships of Equation (3.20) are helpful in understanding the effects of e_{max}. The value e_{max} can be selected by specification or engineering judgment. In either case, Equation (3.2) allows for intuitive choices of e_{max}. For example, if e_{max} is chosen as 15%, then the training algorithm will stop when the average variation between the two estimates exceeds that value. The value of e_{max} is in general the same for each category; however, it can be selected individually for each class. When designing the classifier, the recommended procedure is shown in Table 3.2.

Table 3.2: Weighted-Parzen-window classifier design steps.

1. Select $e_{max} = 0$, and determine the smoothing parameters that minimize classification error[4].
   a. If the training set is small, use the leave-one-out method to estimate the error rate [5:76].
   b. If the training set is large, partition the data into two disjoint sets for training and testing to estimate the error rate [5:76].
2. Choose the value of e_{max} based on design specifications or reduction.
3. Train and evaluate the performance. If performance is satisfactory, implement the device; otherwise go to step 2.

[4] By choosing $e_{max} = 0$, the WPW classifier is equivalent to the Parzen-window classifier (this is proven in Chapter 4). Therefore, the techniques for choosing the smoothing parameter as outlined in references [5], [9], [19], and [20] are valid and should be used. Although the training and classification algorithms allow for selection of different smoothing parameters for each class of data, it is recommended to use a single value for all classes [20:254].

CHAPTER 4
ANALYTICAL RESULTS

The Gaussian distribution function is prominent throughout pattern recognition literature because of its analytical tractability [5:22]. The well-known properties of the Gaussian distribution are extremely helpful when analyzing the WPW classifier. In this chapter, it is shown that the performance of several well-known classifiers can be learned by varying the two system parameters. In particular, the Bayes-Gaussian, minimum Euclidean-distance, Parzen-window, and nearest-neighbor classifiers are derived.

Using Gaussian Windows

To use Gaussian windows, Equation (3.5) is written as

$$g_i(\mathbf{x}) = P(\omega_i)\frac{1}{n_i}\sum_{j=1}^{\hat{n}_i}\frac{1}{\left(2\pi\sigma_i^2\right)^{\frac{d}{2}}}\exp\left[-\frac{1}{2\sigma_i^2}\left(\mathbf{x}-\mathbf{r}_{ij}\right)'\left(\mathbf{x}-\mathbf{r}_{ij}\right)\right]\cdot w_{ij} \qquad (4.1)$$

where $\mathbf{r}_{ij}$ is the jth reference vector ith class and w_{ij} is the corresponding weight, d is the number of dimensions, and $\hat{n}_i$ is the number of reference vectors for the ith class. Note it is customary to replace h_i with σ_i to emphasize the relation of Equation (4.1) to the Gaussian distribution.

Special Case Training Results

This section will show that proper selection of the system parameters σ and e_{max} will result in several well-known classifiers. In what follows, the training algorithm is

shown to be capable of learning Bayes-Gaussian, minimum Euclidean-distance , Parzen-window, and nearest-neighbor performance.

Case 1*: *Bayes-Gaussian Classifier. Given a set of training data for c classes labeled ω_i, $i = 1, \ldots, c$. Choose $e_{max} = \infty$, and σ_i as some suitable number. In this case, σ_i can be different for each class. Consider the effect of e_{max}. Since all error of the training phase is tolerated, a single reference vector will be used to represent each category upon completion of training. Because of the Equation (3.3) used in Step 5 of the training algorithm, the reference vector of each category is exactly the sample mean vector of each category $\hat{\mu}_i$. Consider Equation (4.2) on the final step of training for the ith category:

$$r_{if} = \frac{r_{i1}w_{i1} + r_{i2}w_{i2}}{w_{i1} + w_{i2}}. \tag{4.2}$$

Since this is the combination of the two final reference vectors, $w_{i1} + w_{i2} = n_i$ where n_i is the number of reference vectors at the beginning of training, or equivalently, $w_{i2} = n_i - w_{i1}$, Equation (4.2) can be rewritten as

$$r_{if} = \frac{r_{i1}w_{i1} + r_{i2}(n_i - w_{i1})}{w_{i1} + (n_i - w_{i1})} = \frac{\sum_{j=1}^{n_i} \mathbf{x}_{ij}}{n_i} = \hat{\mu}_i. \tag{4.3}$$

Upon the completion of training, the final reference vector is the sample mean of the training data, $\hat{\mu}_i$, its weight is $w_{if} = n_i$, and $\hat{n}_i = 1$. With this in mind, Equation (4.1) can be rewritten as

$$g_i(\mathbf{x}) = P(\omega_i)\frac{1}{n_i}\sum_{j=1}^{1}\frac{1}{(2\pi\sigma_i^2)^{\frac{d}{2}}}\exp\left[-\frac{1}{2\sigma_i^2}(\mathbf{x} - \hat{\mu}_i)^t(\mathbf{x} - \hat{\mu}_i)\right]\cdot n_i, \tag{4.4}$$

which simplifies to

$$g_i(\mathbf{x}) = P(\omega_i)\frac{1}{\left(2\pi\sigma_i^2\right)^{\frac{d}{2}}}\exp\left[-\frac{1}{2\sigma_i^2}(\mathbf{x}-\hat{\boldsymbol{\mu}}_i)^t(\mathbf{x}-\hat{\boldsymbol{\mu}}_i)\right] \qquad (4.5)$$

By using $\sigma_i^2 I$, where I is the identity matrix, as the covariance matrix of a Gaussian distribution function, Equation (4.5) can be rearranged as shown in Equation (4.6).

$$g_i(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}\left|\sigma_i^2 I\right|^{\frac{1}{2}}}\exp\left[-\frac{1}{2}(\mathbf{x}-\hat{\boldsymbol{\mu}}_i)^t\left(\sigma_i^2 I\right)^{-1}(\mathbf{x}-\hat{\boldsymbol{\mu}}_i)\right]P(\omega_i) \qquad (4.6)$$

Equation (4.6) is the familiar Bayes-Gaussian classifier (see Equation (2.11)) for a class-conditionally independent normal multivariate distribution $\mathbf{x}|\omega_i \sim N[\hat{\boldsymbol{\mu}}_i, \sigma_i^2 I]$. Decision surfaces that result from discriminant analysis are hyperquadric. In the cases that $\hat{\boldsymbol{\mu}}_i$ and σ_i accurately reflect the training data, the WPW classifier is optimal.

Case 2*: *Minimum Euclidean-Distance Classifier. Given a set training data for c classes labeled ω_i, $i = 1, \ldots, c$. Choose $e_{max} = \infty$, $\sigma_i = \sigma$, and $P(\omega_i) = P(\omega) = 1/c$ for all i. Since all error of the training phase is tolerated, a single reference vector will be used to represent each category upon completion of training. Case 1 shows that the resulting discriminant function is given by

$$g_i(\mathbf{x}) = P(\omega)\frac{1}{\left(2\pi\sigma^2\right)^{\frac{d}{2}}}\exp\left[-\frac{1}{2\sigma^2}(\mathbf{x}-\hat{\boldsymbol{\mu}}_i)^t(\mathbf{x}-\hat{\boldsymbol{\mu}}_i)\right]. \qquad (4.7)$$

Given the conditions stated above, this classifier behaves exactly as the minimum Euclidean-distance classifier of Equation (2.22). Furthermore, the decision boundaries are hyperplanes. The following analysis shows why this is true. First, the natural logarithm of Equation (4.7) is written as

$$\tilde{g}_i(\mathbf{x}) = -\frac{1}{2\sigma^2}(\mathbf{x}-\hat{\boldsymbol{\mu}}_i)^t(\mathbf{x}-\hat{\boldsymbol{\mu}}_i) + \ln P(\omega) - \frac{d}{2}\ln 2\pi - d\ln\sigma. \qquad (4.8)$$

Since all of the bias terms that are the same in each category, Equation (4.8) can be rewritten as

$$g_i(\mathbf{x}) = -\frac{1}{2\sigma^2}(\mathbf{x} - \hat{\boldsymbol{\mu}}_i)^t(\mathbf{x} - \hat{\boldsymbol{\mu}}_i)$$

which can be scaled by $2\sigma^2$ giving

$$\tilde{g}_i(\mathbf{x}) = -(\mathbf{x} - \hat{\boldsymbol{\mu}}_i)^t(\mathbf{x} - \hat{\boldsymbol{\mu}}_i). \qquad (4.9)$$

Note that the tilde notation indicates that the discriminant function has been changed, but the classification results have not. Equation (4.9) is the minimum Euclidean-distance classifier as given by Equation (2.22). As shown by Equation (2.23), this case of the WPW classifier results in decision boundaries that are hyperplanes.

Case 3: Parzen-Window Classifier. Given a set training data for c classes labeled ω_i, $i = 1, \ldots, c$. Choose $e_{max} = 0$, and σ as some suitable number. Consider the affect of e_{max}. Since no error can be tolerated during training, there will be no reduction in the reference vector set R_i, therefore, it is equal to the training set X_i. Upon the completion of training for the ith category, n_i reference vectors remain, each of which has a coefficient of 1. Therefore, $w_{ij} = 1$, for $j = 1, \ldots, n_i$, and $\hat{n}_i = n_i$. With this in mind, Equation (4.1) can be rewritten as

$$g_i(\mathbf{x}) = P(\omega_i)\frac{1}{n_i}\sum_{j=1}^{\hat{n}_i}\frac{1}{\left(2\pi\sigma_i^2\right)^{\frac{d}{2}}}\exp\left[-\frac{1}{2\sigma_i^2}\left(\mathbf{x} - \mathbf{r}_{ij}\right)^t\left(\mathbf{x} - \mathbf{r}_{ij}\right)\right]\cdot 1$$

$$= P(\omega_i)\frac{1}{n_i}\sum_{j=1}^{n_i}\frac{1}{\left(2\pi\sigma_i^2\right)^{\frac{d}{2}}}\exp\left[-\frac{1}{2\sigma_i^2}\left(\mathbf{x} - \mathbf{x}_{ij}\right)^t\left(\mathbf{x} - \mathbf{x}_{ij}\right)\right]. \qquad (4.10)$$

Equation (4.10) is the same as Equation (2.33), which is the Parzen-window discriminant function. Another important characteristic of this form of the WPW system is that as

$n_i \rightarrow \infty$, Equation (4.10) approaches the Bayes-optimal classifier. This is a property of the Parzen-window approach coupled with a Bayes strategy. (In this case, it has been assumed that no two reference vectors are identical. If they were identical, they would be combined without introducing error. Therefore, the WPW classifier would still be identical to the Parzen-window classifier, but it would be difficult to analyze.)

Case 4: Nearest-Neighbor Classifier. Given a set training data for c classes labeled ω_i, $i = 1, \ldots, c$. Choose $e_{max} = 0$, $\sigma_i = \sigma$, which is very small, and $P(\omega_i) = P(\omega) = 1/c$ for all i. In this case, as in the above case, reference vector combinations are completely inhibited, and the set of reference vectors R_i is exactly the training set X_i. The discriminant function for this case is given by

$$g_i(\mathbf{x}) = \frac{1}{c}\frac{1}{n_i}\sum_{j=1}^{n_i}\frac{1}{\left(2\pi\sigma_i^2\right)^{\frac{d}{2}}}\exp\left[-\frac{1}{2\sigma_i^2}\left(\mathbf{x}-\mathbf{x}_{ij}\right)^t\left(\mathbf{x}-\mathbf{x}_{ij}\right)\right]. \qquad (4.11)$$

In this case the classifier may not necessarily become optimal as the number of training samples approaches infinity. In fact, the best that one may hope to achieve in this case is known performance. By choosing the value of σ to be very small, the window function becomes very narrow, and only the nearest neighbors of a test sample affect the discriminant function values. In this case, Equation (4.11) approaches the nearest-neighbor (NN) classifier [20:254], but the Parzen-window classifier is still being employed. This approach to NN pattern classification brings to light a very interesting situation. Cover and Hart showed that the NN classifier's error rate can be no worse than twice the Bayes error rate [4] in the large sample case, i.e., as $n_i \rightarrow \infty$. But, remember as $n_i \rightarrow \infty$ the Parzen estimate approaches the true estimate. So, Equation (4.11) becomes optimal if and only if the a priori probabilities of each class are each $1/c$. According to Cover and Hart, the optimal error rate can only occur in "the extreme cases of complete

certainty and complete uncertainty" [4:24]. Therefore, Equation (4.11) is optimal for complete certainty or complete uncertainty; otherwise, it must be strictly greater than the Bayes rate but less than or equal to twice the Bayes rate. Since Equation (4.10) will always provide an optimal classifier in the large sample case, Equation (4.11) would only be used when the number of samples is small and NN classification is desired.

The above analysis showed that several well-known classifiers could be learned by using Gaussian window shapes and selecting the system parameters correctly. The first two derivations rely on the way reference vectors are combined during training and the window function properties. The last two derivations rely on the properties of the Parzen window technique and the error criterion function, i.e., no vectors are combined. Since the WPW algorithm is generally used to reduce the effective size of the training set, i.e., the storage requirements, it should be noted that Cases 3 and 4 are primarily of theoretical importance and are used to bolster the credibility of the algorithm. By making use of the properties of the Gaussian distribution and the Parzen-window classifier, it has been shown that Bayes-Gaussian, minimum Euclidean-distance, Parzen-window, and nearest-neighbor classification can be learned by the WPW algorithm. This is especially useful when the WPW approach is thought of as a black box. In this case, known performance classifiers can always be achieved with a single black box by simply tweaking the system parameters. In this sense, these classifiers can be viewed as special cases of the WPW algorithm. The derivations above are for extreme cases of training, i.e., when a single reference vector remains for each class and when all of the reference vectors remain after training. Other cases are analytically intractable, so it becomes necessary to determine results experimentally.

CHAPTER 5
EXPERIMENTAL RESULTS

The performance of the WPW classifier is described analytically in Chapter 4. There, analyses are applied to special case training scenarios. The capabilities of the WPW technique are difficult to demonstrate analytically when the number of reference vectors is neither 1 nor n. Therefore, the experimental results are used to demonstrate the effectiveness of the WPW classifier for a variety of system parameter choices. This chapter describes the experimental procedure used and results obtained. First, the data is discussed. Then, experimental training results are demonstrated by graphical portraits of its clustering tendencies. Classification results are also presented in the form of decision boundaries and decision error curves. Specifically, the WPW algorithm is compared with those of the Bayes-Gaussian, kNN, and Parzen-window classifiers. Finally, to demonstrate the effects of the two system parameters, a design curve is presented.

The Data

The data used to demonstrate the capabilities of the WPW algorithm was synthesized to be challenging, but also to allow the analytical determination of the Bayesian error rate. Two-dimensional data are used so that the properties may be explored visually, however the WPW algorithm can be used for data of any dimension. An independent Gaussian random variable in two dimensions with unit variance was used to create a two category data set. The first category is bimodal while the second is unimodal

centered between the modes of the first one. The conditional densities are given by Equation 5.1:

$$p(\mathbf{x}|\omega_1) = \frac{1}{2}\left\{\frac{1}{2\pi}\exp\left[-\frac{1}{2}(\mathbf{x}-\mu_{11})'(\mathbf{x}-\mu_{11})\right] + \frac{1}{2\pi}\exp\left[-\frac{1}{2}(\mathbf{x}-\mu_{12})'(\mathbf{x}-\mu_{12})\right]\right\} \quad (5.1.a)$$

and

$$p(\mathbf{x}|\omega_2) = \frac{1}{2\pi}\exp\left[-\frac{1}{2}(\mathbf{x}-\mu_2)'(\mathbf{x}-\mu_2)\right] \quad (5.1.b)$$

where

$$\mu_{11} = [\ 0.0\ 0.0\ ]^t$$

$$\mu_{12} = [\ 5.0\ 5.0\ ]^t$$

$$\mu_2 = [\ 2.5\ 2.5\ ]^t.$$

Figure 5.1 shows the data where Class I is represented by squares while Class II is represented by triangles. The Bayesian error rate, determined analytically, is approximately 5.0%. In all experiments, the a priori probability was assumed to be equal for each category, i.e., $P(\omega_1) = P(\omega_2) = 0.5$. Figure 5.1 shows a subset of the synthesized samples.

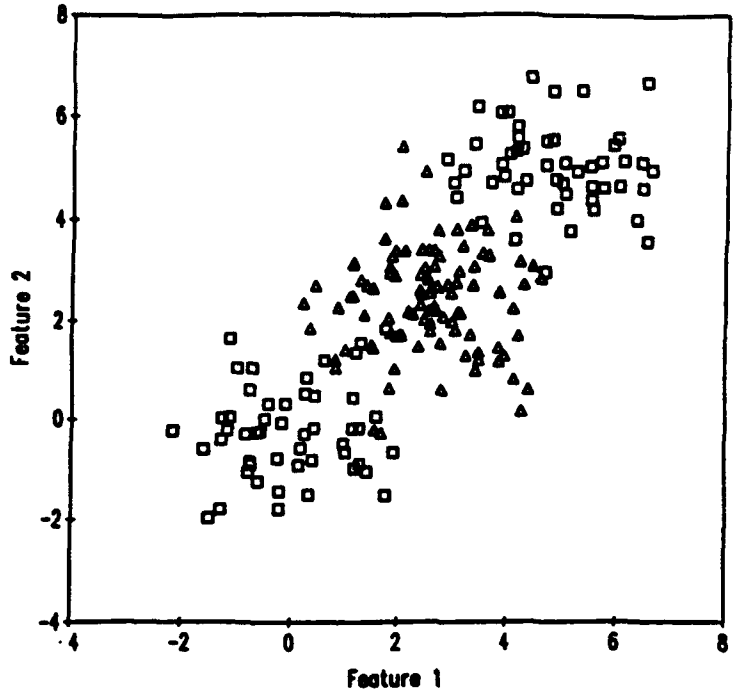

Figure 5.1: Two category sample data.

Training Results

A data set containing 100 samples in each category is used to show the clustering tendencies of the training algorithm as the design parameters vary. In all experiments, the same value of e_{max} and σ is used for each class. The value of σ was selected as outlined in Table 3.2 Step (1.a). Figure 5.2 shows the average error rate for the Parzen-window classifier as a function of σ. Based on Figure 5.2, σ was selected as 1.0 for the following experiments. Figures 5.3-6 show the reference vectors after training for e_{max} equal to 2.5%, 5.0%, 15.0%, and 60.0% respectively. The vectors are shown with their size proportional to their corresponding weights w. Table 5.1 lists the number of reference vectors that resulted after training. It can be seen that whenever the given samples formed compact clusters, they were collapsed into a single vector. On the other hand, those samples that were relatively isolated were preserved. Note that in Figure 5.6, the reference vectors represent the sample mean of each of the modes.

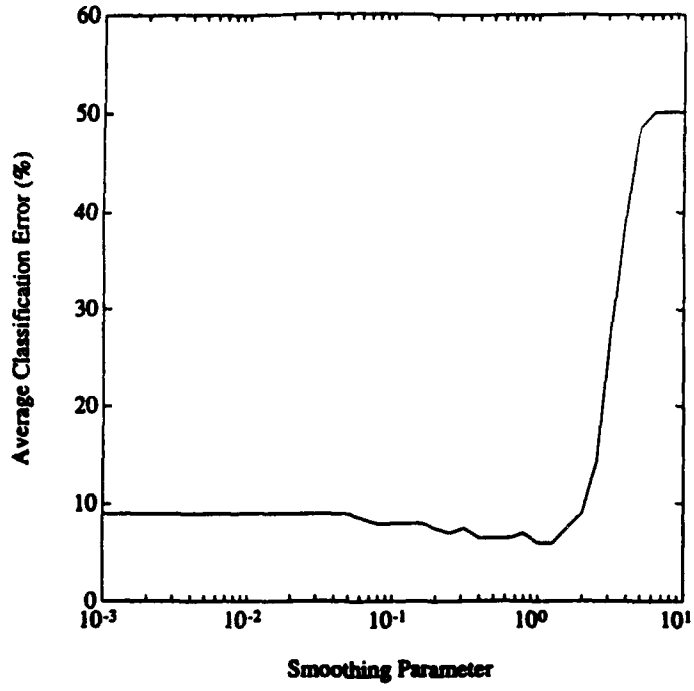

Figure 5.2: Parzen-window classification error vs. smoothing parameter.

Table 5.1: Effect of e_{max} on the number of reference vectors (σ = 1.0).

		$\hat{n}$			
	n	e_{max} = 2.5%	e_{max} = 5.0%	e_{max} = 15%	e_{max} = 60%
Class I	100	32	25	10	2
Class II	100	24	10	6	1

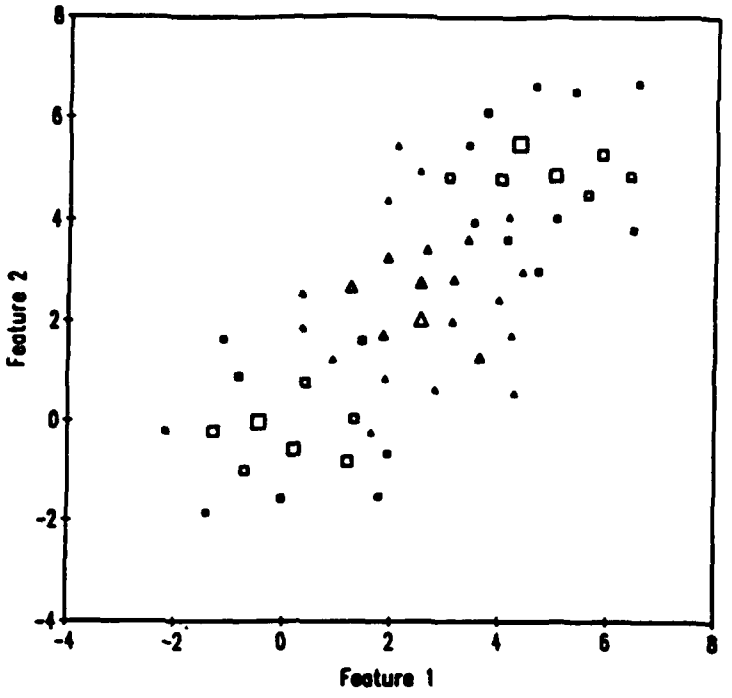

Figure 5.3: Reference vectors after training (e_{max} = 2.5%, σ = 1.0).

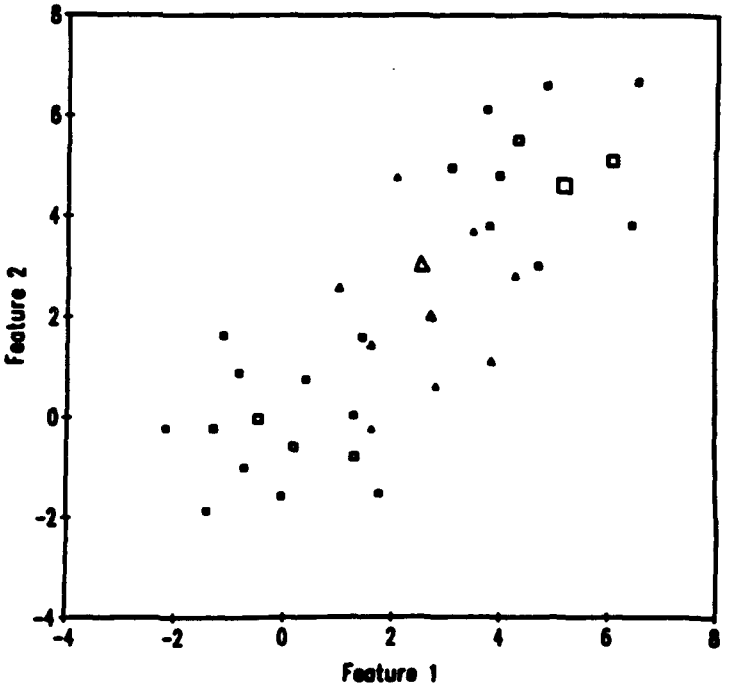

Figure 5.4: Reference vectors after training (e_{max} = 5.0%, σ = 1.0).

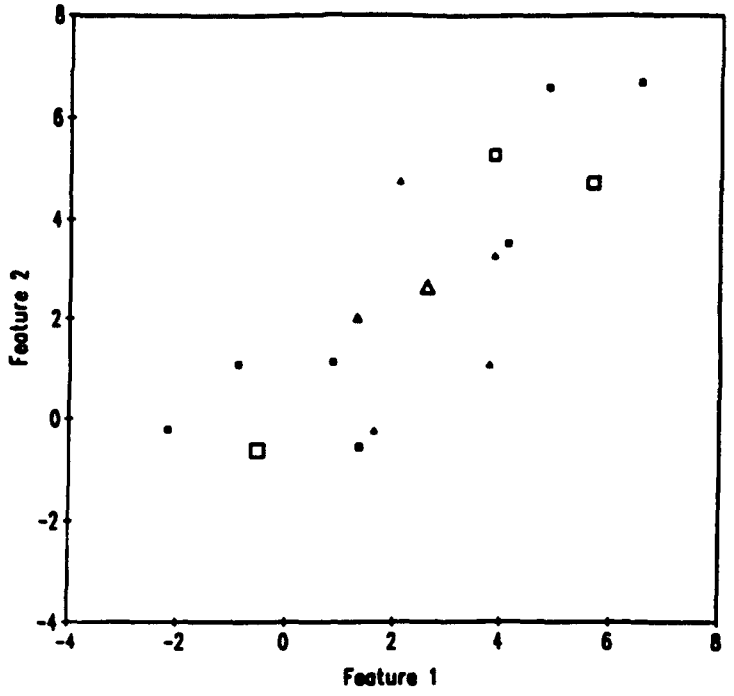

Figure 5.5: Reference vectors after training (e_{max} = 15.0%, σ = 1.0).

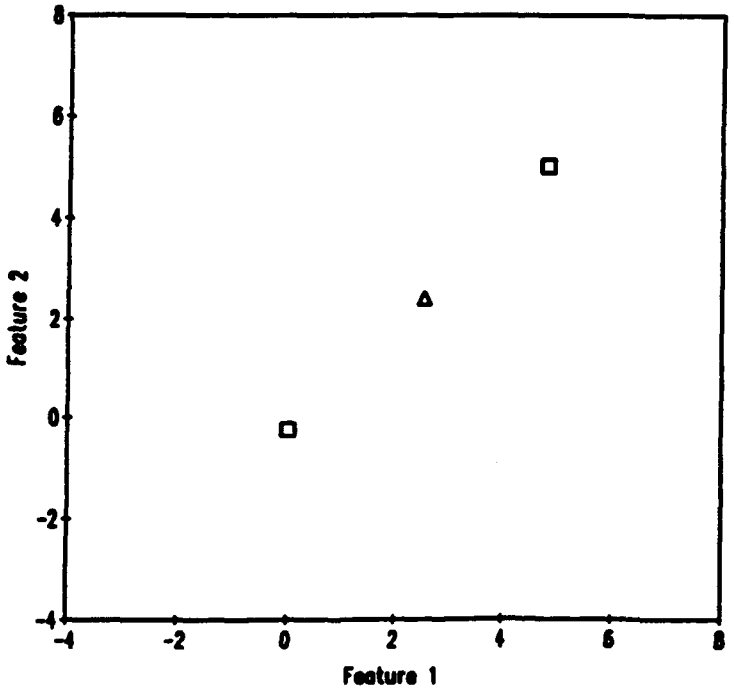

Figure 5.6: Reference vectors after training (e_{max} = 60.0%, σ = 1.0).

Classification Results

Decision boundary graphics provide an excellent tool for visual analysis of classifier performance. The WPW algorithm was compared to the Bayes-Gaussian, Parzen-window, and *k*-nearest-neighbor classifiers.

Figures 5.7 shows the decision boundaries for the Bayesian classifier. Figures 5.8-5.11 show the decision boundaries that result from the Parzen-window classifier when the smoothing parameter is 0.1, 0.5, 1.0, and 2.0 respectively. These figures demonstrate the effect of the smoothing parameter. When σ is small, the decision boundaries are much like the NN classifier. As σ increases, the decision boundaries change from complex undulating lines to very straight lines. Figures 5.12-5.15 show the decision boundaries for the 1, 3, 7, and 21 nearest-neighbor rules respectively. Note that the NN decision rule results in a decision boundary that is jagged and very specific to the training data. However, when the 3NN and 7NN rules are used, the decision boundary becomes smoother and more general. The 21NN rule's decision boundary is smoothest, resulting in the most general of the nearest-neighbor classifiers shown.

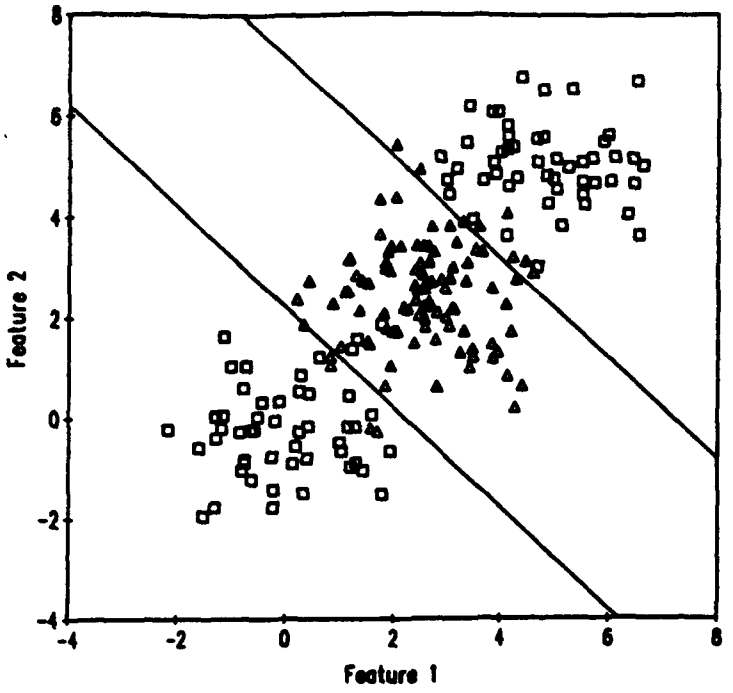

Figure 5.7: Bayesian decision boundaries.

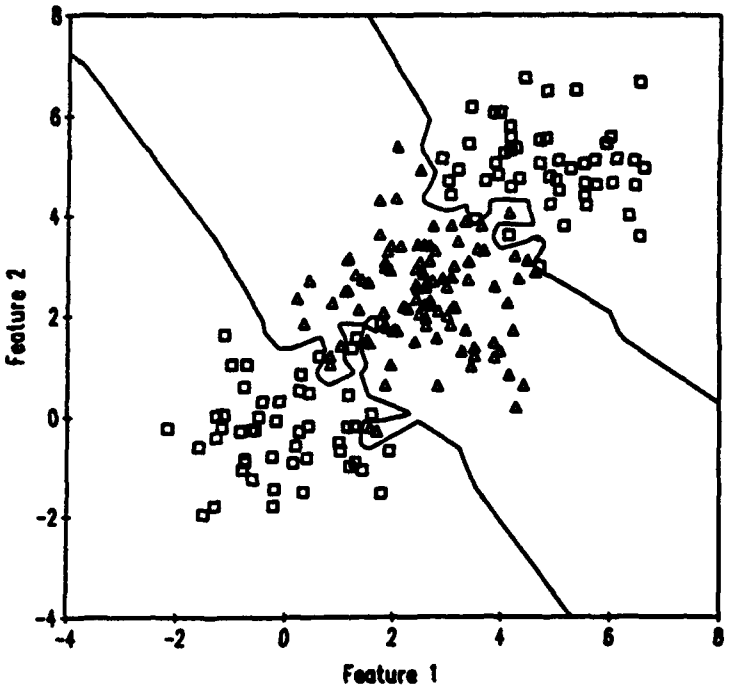

Figure 5.8: Parzen-window decision boundaries ($\sigma = 0.1$).

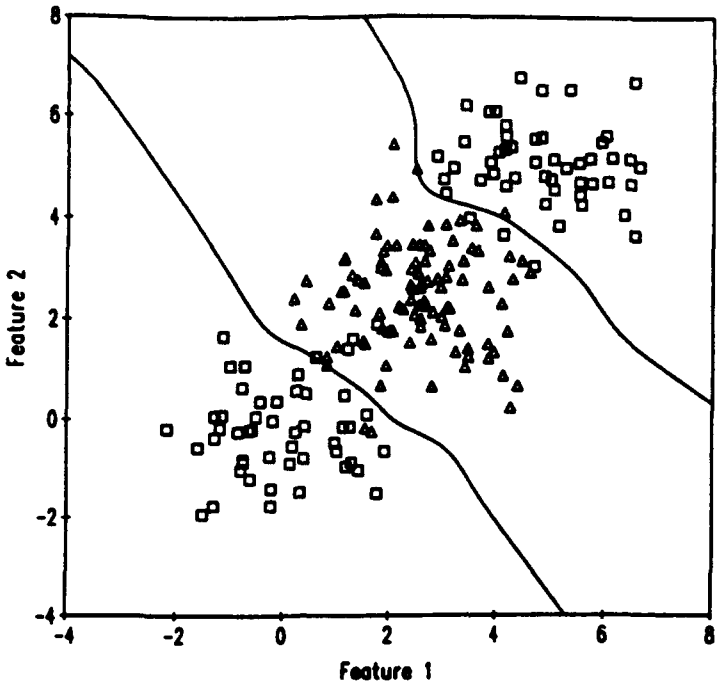

Figure 5.9: Parzen-window decision boundaries ($\sigma = 0.5$).

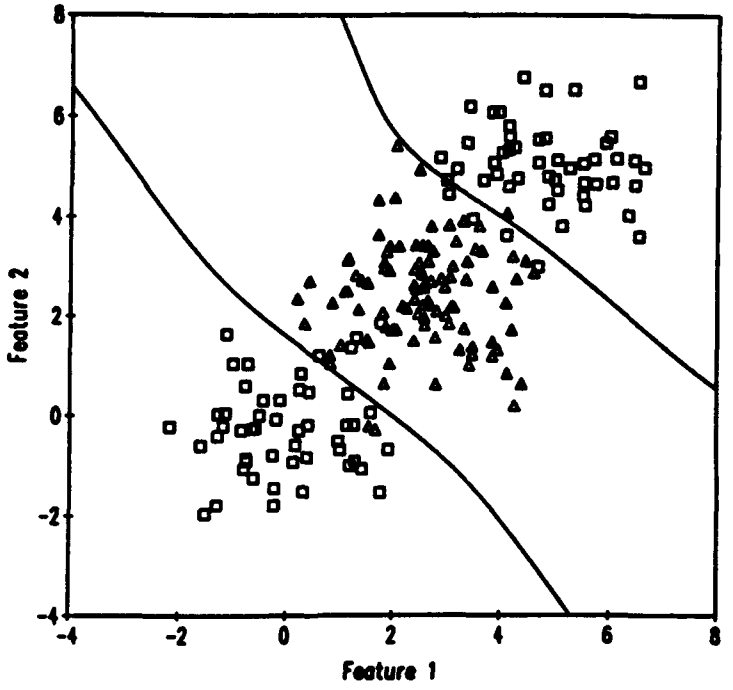

Figure 5.10: Parzen-window decision boundaries ($\sigma = 1.0$).

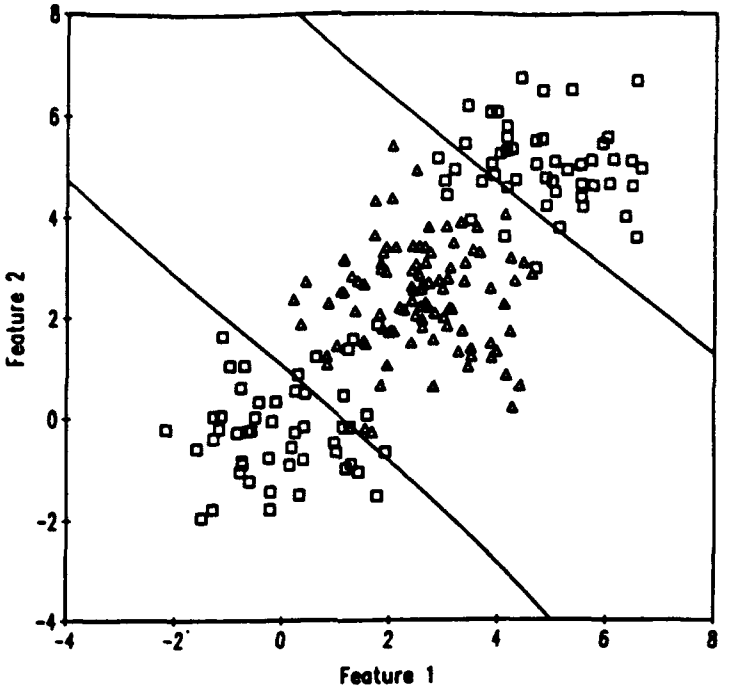

Figure 5.11: Parzen-window decision boundaries ($\sigma = 2.0$).

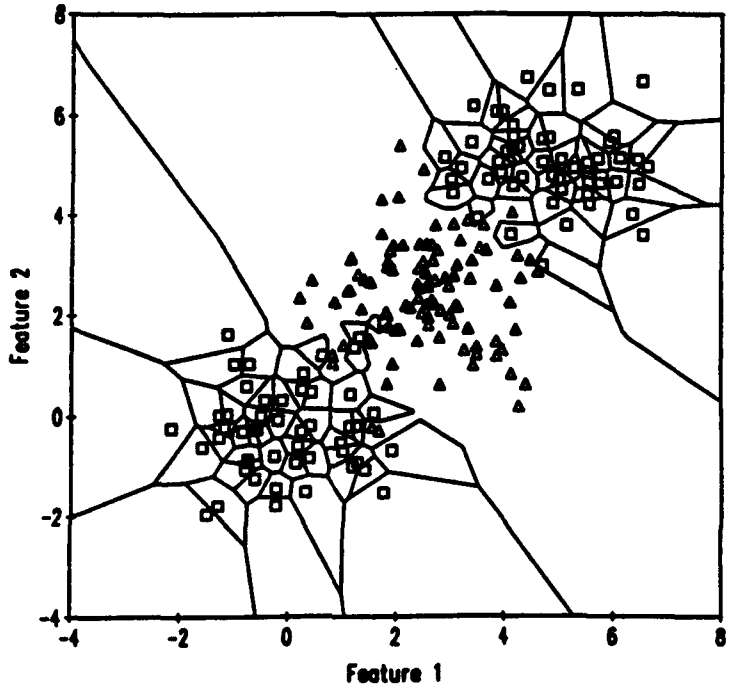

Figure 5.12: Decision boundaries for NN classifier.

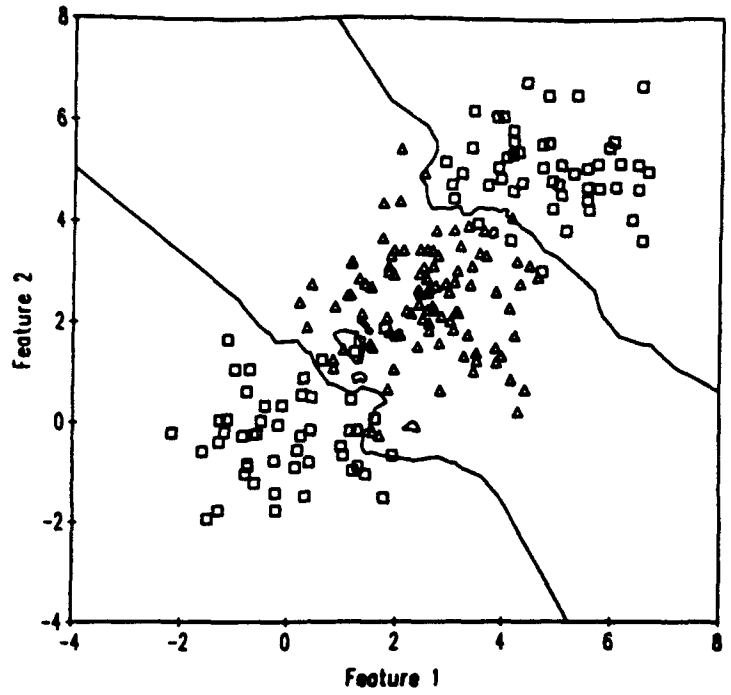

Figure 5.13: Decision boundaries for 3NN classifier.

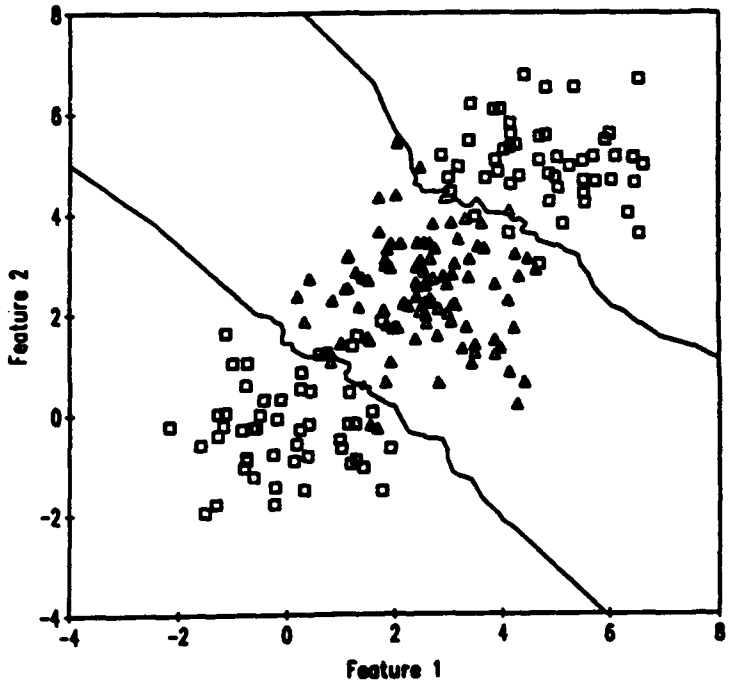

Figure 5.14: Decision boundaries for 7NN classifier.

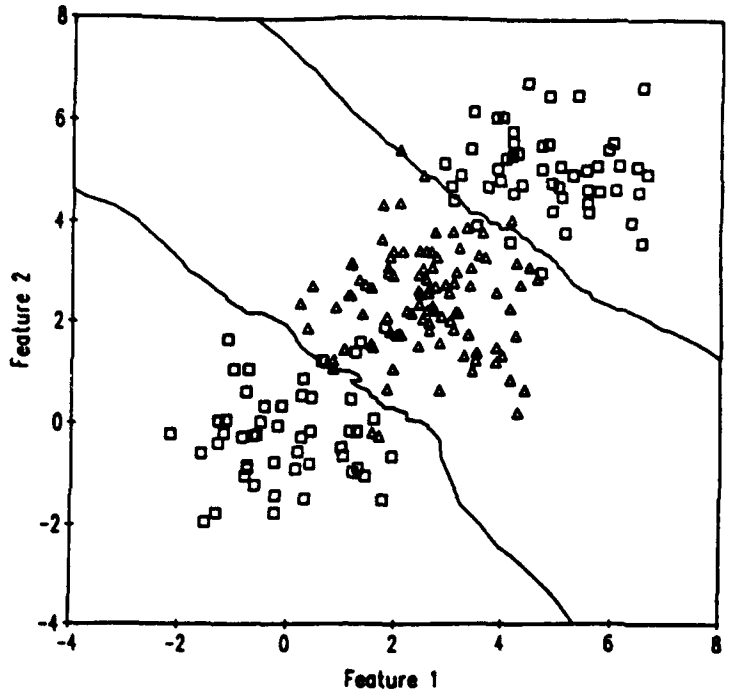

Figure 5.15: Decision boundaries for 21NN classifier.

To demonstrate the WPW classifier graphically, decision boundaries were plotted. The classifier was designed for $\sigma = 1.0$ and varying e_{max}. The WPW decision rule produced the decision boundaries shown in Figures 5.16-5.19 for $e_{max} = 2.5\%$, 5.0%, 15.0%, and 60.0% respectively. The decision boundaries are nearly the same as those produced by the Parzen-window decision rule shown in Figure 5.10 and similar to the decision boundaries of the 21NN rule shown in Figure 5.15. The close relationship to the Parzen-window decision boundaries of Figure 5.10 is expected since the WPW algorithm tries to maintain the Parzen estimate while reducing the effective size of the training set. Note that the decision boundaries are very similar to those of Figure 5.10 even though the reference vectors represent only 28%, 17.5%, 8.0%, and 1.5% of the original training samples. This represents a significant storage reduction while maintaining excellent performance. Note that the decision boundaries of Figure 5.19 are nearly optimal. The above analysis presents a visual comparison of several classifiers. The Bayes classifier is optimal, but requires knowledge of the data's structure a priori. On the other hand, the nonparametric classifiers do not require this knowledge but may require excessive storage and computation time during the classification stage. As seen by the figures, the nonparametric classifiers were capable of achieving excellent results. In particular, the WPW classifier performed as well as the others with the bonus of requiring fewer reference vectors and hence less memory and computational time during classification.

To compare the WPW algorithm another way, the total error rate as a function of the data set size was calculated for the Parzen-window, *k*-nearest-neighbor, and WPW classifiers. In this case, the number of training samples was as large as the number of test samples. Figures 5.20, 5.21, and 5.22 show the error rates. The error shown was calculated by training and then testing with different but equal size sets. The curves show that the WPW's performance is excellent when compared to the others.

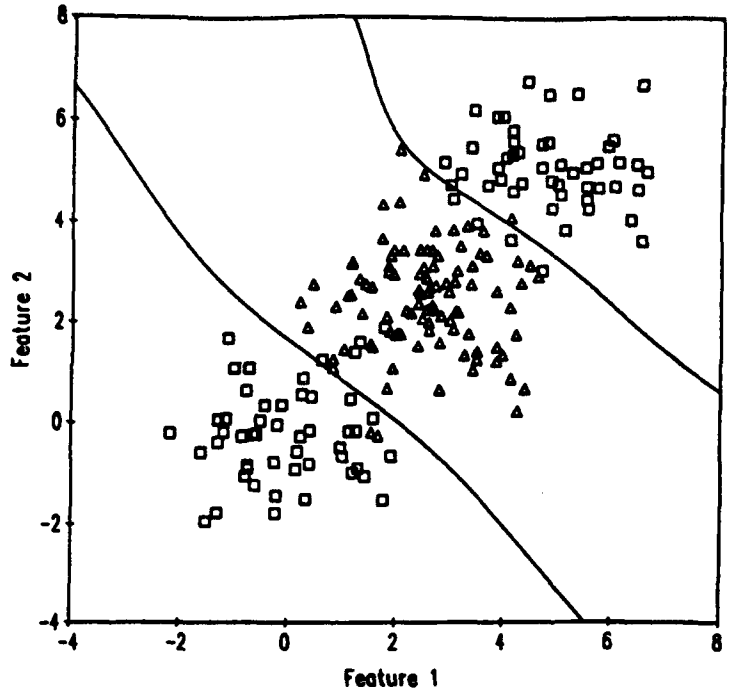

Figure 5.16: Weighted-Parzen-window decision boundaries (e_{max} = 2.5%, σ = 1.0).

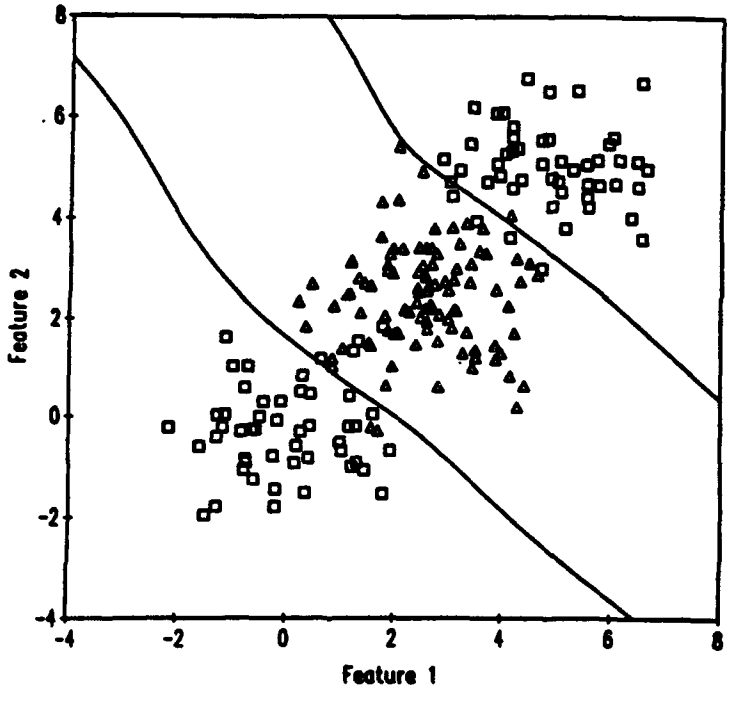

Figure 5.17: Weighted-Parzen-window decision boundaries (e_{max} = 5.0%, σ = 1.0).

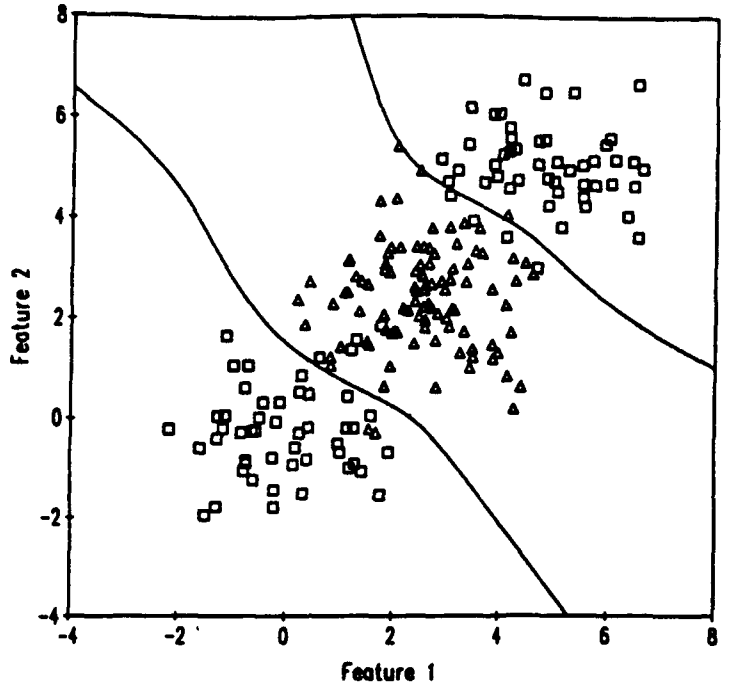

Figure 5.18: Weighted-Parzen-window decision boundaries (e_{max} = 15.0%, σ = 1.0).

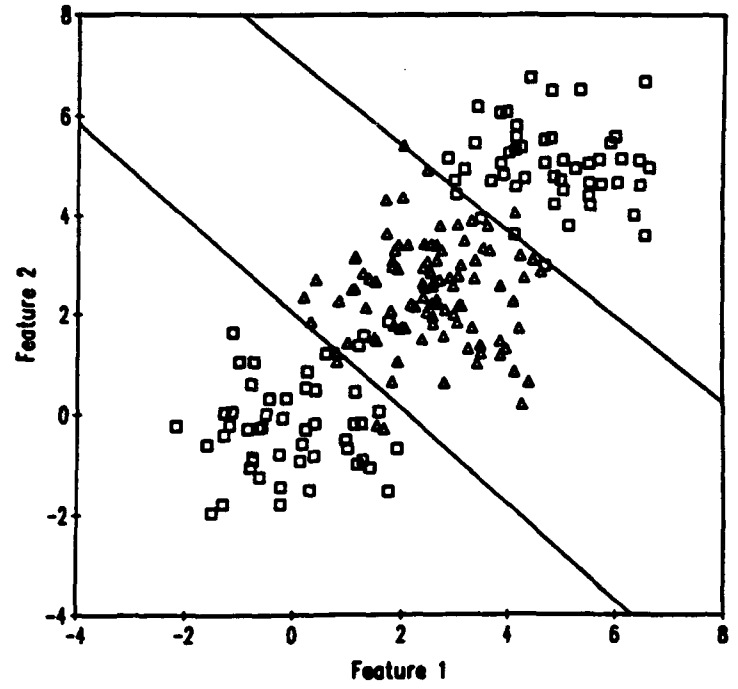

Figure 5.19: Weighted-Parzen-window decision boundaries (ϵ_{max} = 60.0%, σ = 1.0).

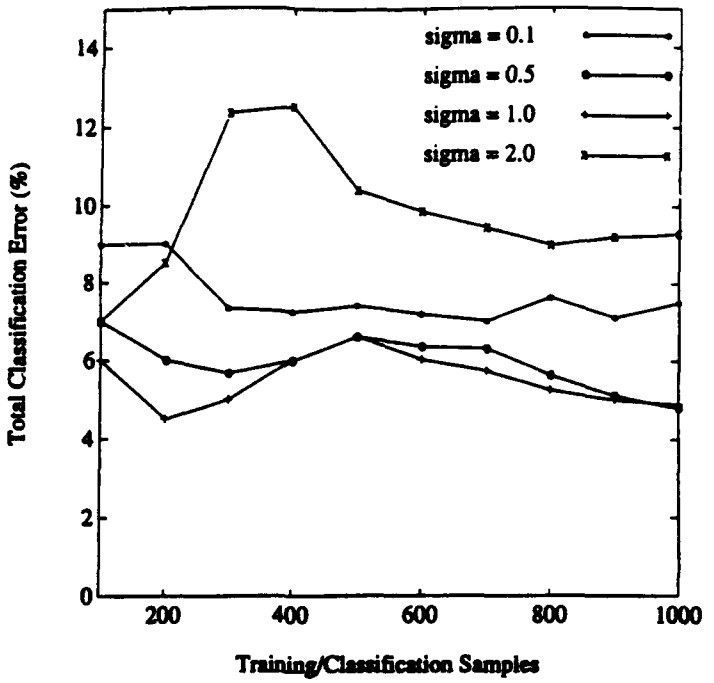

Figure 5.20: Parzen window classification error for various smoothing parameters.

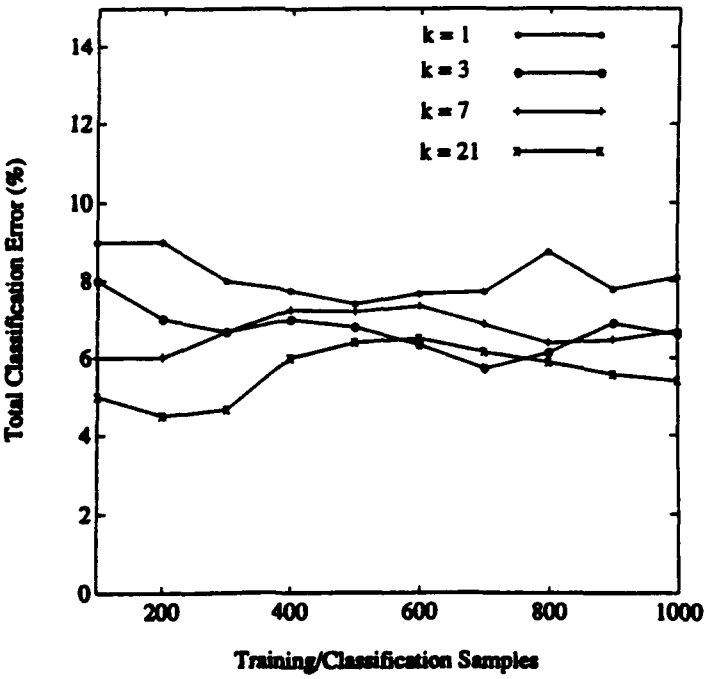

Figure 5.21: Classification error for *k*-nearest-neighbor.

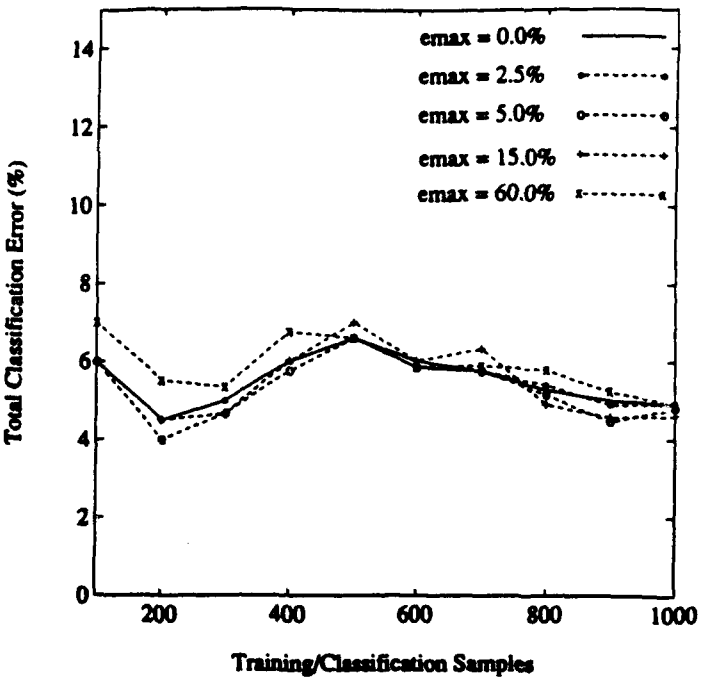

Figure 5.22: Weighted-Parzen-window classification error.

Parameter Design Curve

The selection of the parameters e_{max} and σ have a direct impact on the performance of the system. The effect of their different values is summarized in the curves given by Figure 5.23. These curves were constructed by training the system with 100 samples of each class for different values of e_{max} and σ. For each of these training phases, the average error rate and average total of the reference vectors, $\hat{n}_{total}$, were obtained by using the leave-one-out technique [5:76]. Thus, a point in a curve of Figure 5.23 corresponds to a pair of values (e_{max}, σ), and it is given by the corresponding error rate in the classification stage and $\hat{n}_{total}$. For the data used in this experiment, it was observed that as the allowed error e_{max} increased, the number reference vectors decreased and the classification error increased. Therefore, in this case, there is a trade-off between the classification error rate and the total number of reference vectors. However, given a desired level of performance (in terms of the desired classification rate and time and memory requirements) the curves in Figure 5.22 provide the required values of the parameters e_{max} and σ. This experiment required significant computing resources. However, if they are available, the WPW system can easily be designed by creating curves such as those in Figure 5.23 for many values of σ. Once the curves are plotted, the designer simply chooses the parameters that provide the smallest error rate and the smallest required reference vectors. Such a design scheme can be implemented by the algorithm shown in Table 5.2. Note that WPW classifier design is deterministic if sufficient computing resources are available, i.e., the system parameters can be found that yield the desired performance.

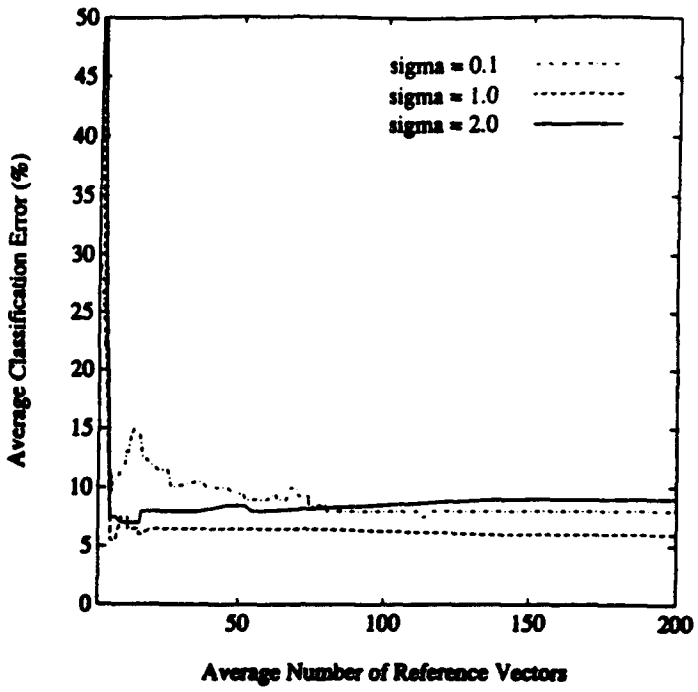

Figure 5.23. Parameter design curve.

Table 5.2: Deterministic weighted-Parzen-window classifier design.

1. Initialize $\sigma = [\ \sigma_1, \ \ldots, \sigma_j\ ]^t$, where j can be any reasonable positive integer, initialize $k = 1$, where $1 \leq k \leq j$, and choose Δ very small. (The smoothing parameter and e_{max} are assumed the same for each class to simplify this procedure. See reference [20:254] for selection of the smoothing parameter.)
2. Initialize $e_{max} = 0.0$ and select σ_k.
3. Use the parameters (e_{max}, σ_k) for WPW training.
4. Estimate and store the total error rate and total number of reference vectors for all classes, $\hat{n}_{total}$.
   a. If the training set is small, use the leave-one-out method when estimating the error rate [5:76] and find the average number of reference vectors.
   b. If the training set is large, partition the data into two disjoint sets for training and testing to estimate the error rate [5:76] and find $\hat{n}_{total}$.
5. IF ($\hat{n}_i > 1$ for any i) THEN

   $e_{max} \leftarrow e_{max} + \Delta$

   go to Step 3

   ELSEIF ($\hat{n}_i = 1$ for all i) THEN

   $k \leftarrow k + 1$

   go to Step 2

   ELSE ($k > j$) Then

   stop

   END.
6. Plot classification error vs. $\hat{n}_{total}$ and choose the parameters (σ_k, e_{max}) that meet or exceed design specifications for storage and classification error.

CHAPTER 6
CONCLUSION

In this chapter, a summary of the thesis is presented. Then, future research goals are discussed. Finally, key research results are highlighted.

Summary

This thesis presents a novel pattern recognition approach, named Weighted Parzen Windows (WPW). In Chapter 2, several pattern recognition concepts and techniques are discussed. They are fundamental to the formulation and analysis of the WPW technique. Techniques discussed in Chapter 2 are the Bayes, minimum Euclidean-distance, Parzen-window, and nearest-neighbor classifiers. It is shown that Bayes-Gaussian classifier is quadratic in the general case and linear when the covariance matrices are equal for each class. Two minimum distance classifiers are presented in Chapter 2. It is shown that the Mahalanobis-distance classifier is quadratic, and the Euclidean-distance classifier is linear. These classifiers are the first of the nonparametric techniques reviewed, i.e., the techniques that do not require a priori knowledge of the density. The next technique is the Parzen-window density estimate. This technique is used to estimate a density function. The Parzen-window classifier is asymptotically optimal when used in a Bayes strategy. The final technique presented in Chapter 2 is the *k*-nearest-neighbor (*k*NN). In Chapter 3, The weighted-Parzen-window (WPW) technique is presented. This technique uses a nonparametric supervised learning algorithm to estimate the underlying density function for each set of training data. The WPW training algorithm quantizes vector space based on a probability space error criterion. It is a second order approximation since it is an

estimate of the Parzen estimate. Classification is accomplished by using the WPW density estimate in a Bayes strategy. In Chapter 3, it is proven that the WPW training algorithm is stepwise optimal. Also, it is shown that the number of distance calculations required is $O(n^3)$ for training and $O(n)$ for classification. Also, an algorithm for WPW classifier design is presented. In Chapter 4, it is shown that several well-known classifiers can be learned by using Gaussian window shapes and selecting the system parameters correctly. By making use of the training algorithm properties, it is shown that Bayes-Gaussian, minimum Euclidean-distance, Parzen-window, and nearest-neighbor classification can be learned by the WPW algorithm. Analytical results are for the cases when a single reference vector remains after training or when all reference vectors remain after training. Other cases are analytically intractable so it was necessary to determine results experimentally. In Chapter 5, experimental results are reported to demonstrate the performance of the WPW algorithm as compared to traditional classifiers.

Future Research Efforts

This research has focused on the development and analysis of the WPW approach as a statistical classifier. Therefore, statistical pattern recognition techniques and philosophies have been paramount to the development and analysis of the novel approach. However, statistical pattern recognition is only one possible use of the WPW technique. It has been designed with other uses in mind. Other applications of the WPW include Artificial Neural Networks (ANNs) and Vector Quantization (VQ). Future research efforts will be directed towards formalizing doctrine for these uses. Also, future research will explore WPW training algorithm refinement.

The WPW as an Artificial Neural Network. The WPW technique can be used in many neural applications. The WPW structure, training algorithm, and performance make it very appealing for Artificial Neural Network (ANN) applications.

The structure of the WPW classifier, in general, is similar to Radial Basis Function (RBF) networks [3], [16], [17]. The WPW classifier with Gaussian windows is a RBF as defined by current literature. However, the RBF training paradigms are theoretically and fundamentally different from the WPW. The training algorithm of the WPW is designed to estimate the density of each class. Therefore, a Bayes approach can always be employed. In general, RBFs do not estimate class densities. The Probabilistic Neural Network (PNN) [24] does try to estimate class densities, however, there are no provisions for reducing the storage requirements.

Many different learning paradigms exist for ANN training. Training usually involves iteratively updating synaptic weights. One such ANN is the error-back-propagation (EBP) technique of D. E. Rumelhart et al. [21]. Until recently, the structure of many ANNs were designed using ad hoc techniques [2]. However, N. K. Bose et al., have presented a technique by which the structure and weights are trained simultaneously [2]. In this technique, Voronoi diagrams (VoDs) are used to design the weights and structure of a feed-forward network. This classifier is trained without the use of parameters. The WPW classifier also trains both the weights and the structure of a feed-forward network. In the WPW technique, the number of reference vectors, i.e., the structure, and values of the weights are determined by selection of two parameters.

Self-organizing maps (SOMs) are popular among neural techniques because of their self-organizing properties. However, they are not generally used directly for pattern classification. Recently, J. A. Kangas et al., reported that SOMs can be used as a preprocessing step for ANNs [13]. Specifically, SOMs can be used to generate

codebooks for Learning Vector Quantization (LVQ) techniques. The WPW algorithm can also be used in this capacity, i.e., it can preprocess the neurons.

The WPW technique can be used for many neural applications; therefore, future research will emphasize these applications. In particular, the neural structure for training will be developed. Also, comparisons will be made with various ANNs. In particular, comparisons will be made with the following techniques: VoD-based classifier [2], EBP [21], LVQ [14], [15], LVQ2 [14], [15], and the PNN [24]. Comparisons will be made with the structure, philosophy, and performance of each of these techniques.

The WPW as a Vector Quantizer. Vector Quantization (VQ) is used for speech and image data compression. Data are compressed and transmitted serially to a receiver where they are decompressed. Compression is accomplished by using a codebook to map the transmission data into one of the codebook vectors. This mapping *quantizes* the transmission data. Reference [1] is an excellent source for further details if necessary. Generation of a good codebook is the main problem in VQ. The WPW training algorithm is directly applicable to codebook generation. Image codebooks can be generated by the WPW algorithm for gray-level and color images. Future research will be directed towards mapping the WPW training algorithm into a parallel structure. Also, experiments will be run to test the effectiveness of the WPW algorithm as compared to traditional codebook generation techniques. (Note: in codebook generation, the required number of calculations can be severe with the WPW technique. Calculation reduction refinements are discussed in the following section.)

WPW Refinements. Research efforts to this point have been focused on the introduction of the WPW technique. Mainly, research has been concerned with the fundamental theory of the technique and its validity. Future research, however, will concentrate on refinements that will ensure efficient practical application. This means

reducing the number of calculations required during training and developing the adaptability of the classifier.

The training algorithm was shown to be $O(n^3)$, therefore, an obvious refinement is to decrease the serial calculations necessary. Possible refinements can be made when finding the two closest vectors in Step 4 of the training algorithm, e.g., preprocessing the reference vectors and recursively updating them on every step. Several quick search routines are available to programmers [6], [9], [23], [26]. It may be possible to modify such a routine for the WPW training algorithm. Also, it may be possible to store the WPW density estimate in a table and updating it recursively on each training step to reduce the calculations necessary in Step 7 of the training algorithm. Parallel computation approaches will be explored. The classifier is highly parallel in structure, therefore, future research will involve mapping the classifier into a parallel architecture. With a parallel approach, WPW training is nearly trivial since all distance calculations can be made simultaneously.

The classifier structure must be enhanced by providing a technique to incorporate new exemplars after training, i.e., develop a mechanism for updating. Future research will focus on a technique that simply includes new exemplars with a single window function. Tentatively, this approach will store all new exemplars until the storage capabilities can no longer accommodate them. At which point, the training algorithm can be invoked to combine vectors where possible. Again, this is a tentative plan which must be further developed.

Conclusions

In this thesis, the WPW classifier was introduced. The WPW classifier is a nonparametric supervised learning technique for pattern recognition. This technique assumes nothing about the underlying density function, but approximates the density function. Since the approach of the WPW is to estimate the density of each class, training can be conducted separately, which means that training can be conducted in parallel. This is an important advantage for parallel processing and hardware implementation. Density estimation is an important concept, because the classifier is designed to approximate the minimum risk Bayes classifier. This is the fundamental underlying philosophy of the approach. However, it is often difficult to store all of the training samples available for the Bayesian nonparametric approach. This is why the WPW technique has been developed. The WPW technique of training set reduction is unique in that it employs two distortion measures. The first is a vector space distortion measure; the second is a probability space distortion measure. This approach quantizes vector space with respect to probability space. The author is unaware of any techniques of this type. Furthermore, the WPW technique was developed and analyzed with traditional statistical techniques. In Chapter 3, it is proven that the WPW training algorithm is stepwise optimal. Also, the computational complexity is discussed. In Chapter 4, the WPW is shown to be functionally equivalent to several well-known classifiers. Chapter 5 gives experimental results.

In Chapter 3, two important results were developed--stepwise optimization and computational complexity. The training algorithm of the WPW is stepwise optimal. This means that each training step introduces the smallest quantization error possible. Also, in Chapter 3, computational complexity is derived. It is shown that the serial distance

calculations necessary are $O(n^3)$ for training and are $O(n)$ for classification. Although the training algorithm is $O(n^3)$, the classification algorithm is very fast, $O(n)$. Furthermore, the training algorithm will converge to a solution after a finite number of training steps. This is an important result when considering that some neural techniques may never converge to a solution.

Chapter 4 presents several important analytical results. In this chapter, it is shown that Bayes-Gaussian, minimum Euclidean-distance, Parzen-window, and nearest-neighbor classification can be learned by the WPW algorithm. This is especially useful when the WPW approach is thought of as a black box. In this case, known performance classifiers can always be achieved with a single black box by simply tweaking the system parameters. In this sense, the above traditional classifiers can be viewed as special cases of the WPW algorithm. Chapter 4 also shows that Bayes performance can be achieved in certain cases. This is of fundamental importance because the Bayes error rate is the theoretical bound.

In chapter 5, experimental results show that the WPW classifier is comparable if not superior to some traditional techniques. Decision boundary graphics and error curves were used to show that the WPW approach reduces the effective size of the training data without introducing significant classification error.

The WPW technique is a very powerful statistical pattern recognition approach. Excellent performance was demonstrated theoretically and experimentally. The WPW technique encompasses many important concepts from statistical pattern recognition. However, the use of the WPW technique should not be limited to statistical pattern recognition. The WPW can be used in neural applications as well as codebook generation for vector quantization. These applications are the subject of current and future research.

REFERENCES

[1] H. Abut, ed., *Vector Quantization*, NY, IEEE Press, 1990.

[2] N. K. Bose and A. K. Garga , "Neural Network Design Using Vornoi Diagrams" *IEEE Transactions on Neural Networks*, Vol. 4, No. 5, pp. 778-787, September 1993.

[3] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks," *IEEE Transactions on Neural Networks*, Vol. 2, No. 2, pp. 302-309, March 1991.

[4] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," *IEEE Transactions on Information Theory*, Vol. IT-13, No. 1, pp. 21-27, January 1967.

[5] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, NY, John Wiley & Sons, 1973.

[6] J. A. Friedman, F. Baskett, and L. J. Shustek, "An algorithm for Finding Nearest Neighbors," *IEEE Transactions on Computers*, Vol. C-24, pp. 1000-1006, October 1975.

[7] K. Fukunaga and J. M. Mantock, "Nonparametric Data Reduction," *Transactions IEEE Pattern Analysis and Machine Intelligence*, PAMI-6, pp. 115-118, 1984.

[8] K. Fukunaga and R. R. Hayes, "The Reduced Parzen Classifier," *Transactions IEEE Pattern Analysis and Machine Intelligence*, PAMI-11, pp. 423-425, 1989.

[9] K. Fukunaga, *Statistical Pattern Recognition*, 2d ed., San Diego, Academic Press Inc., 1990.

[10] E. S. Gelsema and L. N. Kanal, (Eds.), *Pattern Recognition and Artificial Intelligence*, New York, North-Holland, 1988.

[11] M. Hammerstrom, "Neural Networks at Work," *IEEE Spectrum*, NY, IEEE Inc., pp. 26-32, June 1993.

[12] P. E. Hart, "The Condensed Nearest Neighbor Rule," *IEEE Transactions on Information Theory*, IT-14, pp. 515-516, 1968.

[13] J. A. Kangas, T. Kohonen, and J. T. Laaksonen, "Variants of Self-Organizing Maps," *IEEE Transactions on Neural Networks*, Vol. 1, No. 1, March 1990.

[14] T. Kohonen, G. Barna, and R. Chrisley, "Statistical Pattern Recognition with Neural Networks: Benchmarking Studies," *IEEE International Conference on Neural Networks*, San Diego, pp. I-61 - I-68, 1988.

[15] T. Kohonen, *Self-Organization and Associative Memory*, 3rd ed., Berlin, Springer-Verlag, 1989.

[16] J. Moody and C. Darken, "Fast-Learning in Networks of Locally Tuned Processing Units," *Neural Comput.*, Vol. 1, No. 2, pp. 281-294, 1989.

[17] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels, "On the Training of Radial Basis Function Classifiers," *Neural Networks*, Vol. 5, pp. 595-603, 1992.

[18] N. J. Nillsson, *The Mathematical Foundations of Learning Machines*, intro., Sejnowski, T. J. and White, H., CA, Morgan Kauffman, 1990.

[19] E. Parzen, "On Estimation of a Probability Density Function and Mode," *Ann. Math. Stat.*, 33, pp. 1065-1076, September 1962.

[20] S. J. Raudys and A. K. Jain. "Small Sample Size Effects in Statistical Pattern Recognition: Recommendations for Practitioners," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, No 3, pp. 252-264, March 1991.

[21] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing, Volume 1: Foundations*, MA, MIT Press, 1986.

[22] R. J. Schalkoff, *Pattern Recognition: Statistical, Structural and Neural Approaches*, NY, John Wiley & Sons, Inc., 1992.

[23] I. K. Sethi, "A Fast Algorithm for Recognizing Nearest Neighbors," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-11, No. 3, pp. 245-248, March 1981.

[24] D. F. Specht, "Probabilistic Neural Networks and the Polynomial Adaline as Complementary Techniques for Classification," *IEEE Transactions On Neural Networks*, Vol. 1, No. 1, pp. 111-121, March 1990.

[25] D. L. Wilson, "Asymptotic Properties of the Nearest Neighbor Rules Using Edited Data Sets," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-2, pp. 408-420, 1972.

[26] T. P. Yunck, "A Technique to Identify Nearest Neighbors," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-6, No. 10, pp. 678-683, October 1976.